PROOF OF PURCHASE

Software Giveaway with Purchase of this Magazine

See back cover

HOME COMPUTER® magazine

FOCUSING EXCLUSIVELY ON • APPLE • ATARI • COMMODORE • IBM • TEXAS INSTRUMENTS

Vol. 5 No. 6

$3.50 in USA
$4.50 in Canada

FREE SOUNDSHEET
Of Computer-Orchestrated Synthesizers & Drum Machines

FlashMail

Computer-Help For ZIPier Letters



Featuring A Ready-To-Use Electronic Postman for:

- Database Construction
- Systematic Search & Sorts
- Personalized Mass Mailings

Special Apple, Atari, Commodore, IBM, & TI Software:

- Assembler For A Song
- Living Cell Bio-Simulation
- File-Shuffling Management Tricks
- Serf City Econo-Fun

Contains **52** Type-In Programs!

—Reviews Galore:

★ Financial Recipe Round-Up
★ Speed-Reading Illusions
★ Down-To-Earth & Blue-Sky Edu-Tools
★ Major Memory Expanders

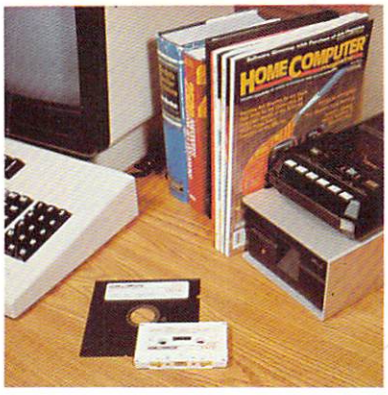
0 12509 06 477465



MISSING ANY VALUABLES?

If you're missing any back issues of HOME COMPUTER™ magazine you're missing more than you'll ever know . . .

Having each issue of *Home Computer Magazine* readily at hand provides you with direct access to a valuable reference library of home computer knowledge —unequaled anywhere!



A valuable reference library of each *Home Computer Magazine* issue is the One Essential Peripheral™ for your home computer.

Back Issues of HCM's program service —ON DISK™ or ON TAPE™ are also available.

Collect all programs from each magazine issue on a ready-to-RUN quality floppy disk or cassette tape available in separate versions for Apple, Atari, Commodore, IBM, and Texas Instruments.



ON DISK™ and ON TAPE™ are the convenient, accurate and affordable ways to save hundreds of typing hours.

SPECIAL CLOSE-OUT SALE!

ALL 4 BIG ISSUES SHOWN ON THE OPPOSITE PAGE
(Vol. 4 No. 1, Vol. 4 No. 2, Vol. 4 No. 3, Vol. 4 No. 4)
DELIVERED TO YOUR DOOR FOR ONLY $5.
USE REGULAR BOUND-IN ORDER FORM TO WRITE-IN SPECIAL SALE ORDER.
No phone orders less than $10 accepted. Limited-Time Offer—While Supplies Last.

"Safeguard" Your Home Computer Knowledge— Order Valuable Back Issues Today!

To Order, Use Bind-In Card at Center of Magazine.

HCM BACK ISSUES

. . . and discover what's in store for you— an extraordinary resource value!

VERSIONS SUPPORTED:

Machine	Media
APPLE II Family (A)	ON DISK™
Atari (At) (coverage commenced with issue 5.5)	ON DISK™/ON TAPE™
COMMODORE 64 (C)	ON DISK™/ON TAPE™
IBM PC/PCjr (I)	ON DISK™
TI-99/4A (T)	ON DISK™/ON TAPE™

*** = No ON TAPE™ available, even if normally supported**
TX = Extended BASIC programs only
PCjr = Available for PCjr only

Apple owners: Please note that ON DISK™ Media for *HCM* 4.1-4.3 is in DOS 3.3 format only, and all Apple programs beginning with *HCM* 4.4 are in ProDOS format. All programs will RUN on a 64K Apple II + (with Applesoft BASIC in ROM), an Apple *IIe*, or an Apple *IIc*.

Apple & IBM "clone" owners: Some *HCM* programs may not RUN (without modification) on your machines, because of differences in hardware and/or BASIC interpreters.



Issue 4.1:

Premier Issue * Uncle Larry's Fiddle Tunes * Electronic Sheet Music * Music in Mini Memory * PCjr: A Look Inside the Peanut's Shell * 66 Keys to Graphics Success: A Primer for the Commodore 64 * Have No Fear: Assembly Language Won't Byte, Part 3 * Porsches and other Pipedreams: Computer Assisted Savings * 3DIIe: Apple Graphics in Three Dimensions, Part 1 * Biting Into Your Apple * Don't Be A SlowPOKE * Down Memory Lane: Don't let programmable characters gobble up your memory * Easy As Pie: Apple programming for intricate works of art * Microcomputer Accuracy * What is LOGO? * Lyrical LOGO * LOGO Shoots for the Moon: A lesson in structured problem-solving * Product Reviews * Flak Attack * Slots * Meltdown * Challenging the Tower of Hanoi * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Flak Attack (A,C,I,T)
Air-to-ground battle game
Applesoft 3D (A)
Apple graphics in three dimensions
Tower of Hanoi (A,C,I,T)
An ancient brain teaser
Saving (A,C,I)
Computer-assisted savings plan
LOGO Poet (A,C*, I)
Recursion frees the poet in your console
LOGO Apollo (C*, T)
A lesson in structured problem-solving
Slots (T)
An intriguing Las Vegas simulation
Uncle Larry's Fiddle Tunes (C, I, T)
Play ten beloved fiddle tunes
Music Magic (TX)
"Joy to the World" in harmonious BASIC
Music Assembler (T)
Assembly language simplifies composition
Autosprite (C)
Routines to keep your graphics lively
Meltdown (TX)
Debug the reactor and save the world



Issue 4.2:

Graphics * Sea of States * San Francisco Tourist * Building Your Character: A Graphics Editor for the VIC-20 * Quick Pixel Tricks: A Graphics Editor for the C-64 * Follow the Bouncing Ball: On the rebound with graphics fundamentals * 3DIIe: Apple Graphics in Three Dimensions, Part 2 * Double Your Color, Double Your Fun: Sprites try on a layered look * Musical Mystery Words * Matrix Muncher * Elementary Addition and Subtraction for the VIC-20 * IBM Animation: Controlling the pallet on the PCjr * Jr. Sounds Off: Access Jr's Special Sound Enhancements * The Electronic Home Secretary * Files in LOGO * LOGO Spans the Generation Gap: A review of Commodore LOGO * FROGO: LOGO Invades the Arcade * Product Reviews * Tablut * Cannibals * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Cannibals (A, C, I, T)
Livingston Stew, I presume?
FROGO (T)
A logical LOGO learning lesson
The Home Secretary (A, C, I, T)
Address & inventory recordkeeping
LOGOFILES (A, C*, I, T)
Access your DATA files in LOGO
Sea of States (C, TX)
State Capitals and dive for booty
Tablut (C, I, TX)
14th-century strategy revisited
Matrix Muncher (C)
Solve unknowns simultaneously
Graphic Editor (C)
Pixel tricks create easeful graphics
Mystery Words (A, I)
Reading between the treble clefs
PCjr Animation (PCjr)
Exploring Junior's graphic modes
Applesoft 3-D IIe (A)
Edit your 3-D graphic shapes



Issue 4.3:

Productivity * Snap-Calc: A Homespun ready-to-use spreadsheet * Bars and Plots: Create colorful graphic charts of your records * Elementary Addition and Subtraction for the 99/4A and C-64: A powerful children's learning tool * Spider Graphics: Spin a colorful web on screen * Convertible for Comfort: Automatically convert your machine-language programs to DATA statements * Programming: The Name of the Game: Designing your own game—a complete tutorial * Colorfun on your VIC-20 * Product Reviews * Binary Forest: Branching out with LOGO * LOGO Flakes: Creative explorations with snowflake designs * Robochase * Cyber-Cipher * Wild Kingdom * Speeder * Boolean Brain * Missile Math * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Snap-Calc (A, C, I, TX)
Home sweet spreadsheet
Robochase (A, C, I)
Run from the rampaging robots
Spider Graphics (A, I)
Spin a myriad of rainbow filaments
Boolean Brain (A, I)
A graphic Adventure inside computers
Wild Kingdom (A, C, I, TX)
Flee ferocious felines
Missile Math (A, C, I, T)
Launching grade-school arithmetic
Binary Forest (A, C*, I)
Branching-out with leafy LOGO trees
Bars & Plots (T)
Color your chart factfully
Cyber-Cipher (T)
Decode correct color combinations
Elem. Addition & Subtraction (C, T)
BASIC preschool arithmetic skill-builder
LOGO Flakes (T)
Snowflakes in June? This must be LOGO
Convertible for Comfort (C)
Machine Language DATA auto-conversion



Issue 4.4:

Computer Sports * IIc: The Core of a New Machine * On the Home Court: Computer Sports Simulation * Razzle Dazzle: Quick Graphics Magic for the 99/4A * Simon Sez: Plug in 114 new BASIC commands to the Commodore 64 * Tax Deduction Filer: A complete tax recordkeeping program convinces you that makes tracking of deductions a breeze * Kaleido Computer: Creating a myriad of mosaic designs on your home computer * Multiplan Medium, Part 8 * Have No Fear: Assembly Language Won't Byte, Part 4 * The RS-232 Interface: Understanding Your Link to the Periphery * One for the Money, Two for the Slow—Adding a Second Drive to the PCjr * Missionary Impossible: A Logic Puzzle in LOGO pits you against hungry Cannibals * Product Reviews * Boolean Brain * Stadium Jumping * Market Madness * Elementary Addition and Subtraction: An arithmetic tutor (for Apple and IBM PC and PCjr systems) * HCM TECH NOTES: Apple, C-64, IBM and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Boolean Brain (C, TX)
A graphic Adventure inside computers
Tax Deduction Filer (A, C, I, TX)
SAVE-ing with your tax deductions
Market Madness (A, C, I, TX)
Exciting Stock market simulation
Stadium Jumping (A, C, I, T)
Horsing around an Olympic Stadium
LOGO Spreadsheet (A, C*, I, T)
And you thought LOGO was kidstuff
Missionary Impossible (A, C*, I, T)
Watch out for Cannibals with LOGO
Elem. Addition & Subtraction (A, I)
BASIC preschool arithmetic skill-builder

And There's More On The Next Page



Issue 4.5:

Building Up Your Software Library * Quiz Construction Set: Create a Quiz or Take a Quiz—a must for students and teachers * Personal Loan Calculator: Find out where your interest lies * Jumping Ahead With Game Programming: A complete game programming tutorial includes a program example * Sketch-64: Joystick graphics with just a flick of the wrist * Simon Sez: New string-related commands explained * Razzle Dazzle: Character manipulation on the 99/4A * Division Tutor: Teaching BASIC math learning skills * Putting The Puzzle All Together: Apple IIc Programming Considerations * Bird Brain * Slither * LOGO Clones: TI Graphics In a Turtle-Shell * Build A LOGO Adventure, Part 1 * Product Reviews * HCM One Liners * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Bird Brain (A, C, I, T)
Keep your fishing feathers dry
Division Tutor (A, C, I, TX)
Expand elementary math skills
Personal Loan Calc (A, C, I, T,)
Find out where your interest lies
Sketch-64 (C)
Use a joystick to draw graphics
Quiz Construction Set:
Quiz-Make/Quiz-Take (A, C, I, T)
Complete tutorial with file examples
Peg Jump (A, C, I, T)
Learn BASIC game programming
Slither (A, C, I, T)
A maze of snake-like proportions
LOGO Clones (T)
TI-Graphics in a Turtle-Shell
LOGO Adventure (A, I, C*)
Pt. 1: Creating interactive fiction



Issue 5.2:

Number Crunching: The Building Blocks of All Computing * It Figures: An equation calculator that'll crunch your numbers accurately * Evacu-Pod: See if you can rescue all the miners in this challenging space game * Switch 'n' Spell: Electronic anagram brain teasers to puzzle over (for children, and adults) * Laserithmetic: Strut your math skill with this space fantasy edu-game * Organizer Reports: An enchancement to print-out your organized thoughts (see The Organizer *HCM* 5.1) * Razzle Dazzle: Tinker with musical sounds, or Play it Maestro! * What is CP/M?: Learn the Basics of Control Programming for Microcomputers * Apple Seedlings: Sorting out your ProDOS Catalog * Commodore Hornblower: Discover what's inside the Commodore 64's SID chip * IBMpressions: Create 3-D surface drawings in BASIC * Field & Screen: A tutorial for using a Data Base System—correctly * Product Reviews * HCM One Liners * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Evacu-Pod (A, C, I, TX)
Miner rescue in space
It Figures! (A, C, I, TX)
A mighty equation calculator
Laserithmetic (A, C, I, T)
Blast aliens with your math skills
Organizer Reports (A, C*, I, TX*)
Print your organized outlines
Switch 'n' Spell (A, C, I, T)
A spelling aid that's fun to boot
Apple Seedlings (A)
Sort your ProDOS catalogs
Commodore Hornblower (C)
Inside the SID chip
IBMpressions (I)
3-D surface drawing in BASIC



Issue 5.4

Time Management: Computer-Assisted Efficiency Comes Home * Run-Day-View: Let your computer streamline your day * Trig-Trix: Use the triangle as a measuring tool * Archeodroid: Participate in a future archeological dig * Mine Over Matter: Hone your business skills in this simulation of uranium mining operations * MAC-ROs: Create custom graphic shapes on Macintosh * IBMpressions: Create your own computer windows * Razzle Dazzle: Explore sound-on-sound recording with this 3-track recorder program * Apple Seedlings: Use a pie chart as a visual aid * Commodore Hornblower: Change filters on the SID chip * Algorithm-A-Tricks: Create invisible ripples * Speeding up a BASIC Program: Part 2 * Build a Logo Adventure: Part 4 * Product Reviews * HCM One Liners * Home Computer Industry Journal * Product News * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Run-Day-View (A, C, I, T)
Organize your daily schedule
Trig-Trix (A, C, I, T)
The triangle makes measurement easy
Archeodroid (A, C, I, T)
Join a future archeological dig
Mine Over Matter (A, C, I, T)
Manage a uranium mine
LOGO Adventure, Pt. 4 (A, C, I)
Apple Seedlings (A)
Creating pie charts
Commodore Hornblower (C)
Changing filters on the SID chip
Apple Tech Note (A)
Exiting error routines
Commodore Tech Note (C)
Merging programs from tape
IBM Tech Note (I)
Using special character graphics

NOTE: Programs for the IBM PC/PCjr will run on the Tandy 1000 with modifications specified on page 130.



Issue 5.1:

Thought Processing: A New Frontier in Home Computing * The Organizer: Store and organize your thoughts * Orbital Defender * Quiz-Print/Quiz-Print Tutorial: This educational enhancement is a tool for use with your Quiz Construction Set (see *HCM* 4.5) * Electronic Backgammon: A modern version of an ancient game of skill * Razzle Dazzle: Screen patterns with graphics characters on the 99/4A * Kors-Elf: An Arcade Typing-Tutor Game * Personal Loan Calculator: Find out where your interest lies * Apple Seedlings: A ProDOS Date-Setting Utility * IBMpressions: Create a beautiful pie chart * Build A LOGO Adventure, Part 2 * LOGO Sailing: A Premier Yachting Event * Simon Sez: Composing music is simple * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Orbital Defender (A, C, I, T)
Split-second battle decisions
Electronic Backgammon (A, C, I, TX)
Pit your pips against the computer
Kors-Elf (A, C, I, TX)
An arcade typing adventure
The Organizer (A, C*, I, TX*)
A versatile Thought Processor
Quiz-Print (A, C, I, T)
Format printouts of your quizzes
Apple Seedling (A)
BASIC utility dates ProDOS files
LOGO Adventure (A, C*, I)
Pt. 2: Creating interactive fiction
Merging Files (C)
Experienced hackers only!
Personal Loan Calc (T)
Find out where your interest lies
Razzle Dazzle (T)
Wormwood your character graphics
LOGO Sailing (T)
Turtles race for the America's Cup
IBMpressions (I)
Create a beautiful pie chart



Issue 5.3

Computerized Budgeting: Featuring a ready-to-use budget processor (Budgetron) * Honing your Geometry skills (Geometrix) * LOGO Adventuring (Build A LOGO Adventure, Pt. 3) * Survive a nuclear plant disaster (Over-Reaction) * Guard the seaways with nuclear submarines (Torpedo Alley) * Turtles race with Zeno's theory (Achilles and the Turtle) * Apple Seedlings: Character graphics on the hi-res screen * Commodore Hornblower: Select waveforms and envelopes from SID * Razzle Dazzle: Multi-layered animation with TI sprites * IBMpressions: Blending sign waves into complex patterns * MAC-ROs: Expanding BASIC on Macintosh * Speeding Up a BASIC Program * Product Reviews * HCM One Liners * Group Grapevine * Product News, * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Budgetron (A, C, I, T)
Budget your income and expenses
Geometrix (A, C, I, T)
Sharpen your geometry skills
Over-Reaction (A, C, I, T)
You're at a nuclear plant's controls
Torpedo Alley (A, C, I, T)
Keep the enemy's ships at bay
Achilles & the Turtle (T)
A LOGO demonstration of Zeno's Theory
LOGO Adventure, Pt. 3 (A, C*, I)
Apple Seedlings (A)
Character graphics in hi-res
Commodore Hornblower (C)
Waveforms & envelopes from SID
Apple Tech Note (A)
Key-in checking routine
IBM Tech Note (I)
Selective keyboard input
Commodore Tech Note (C)
Merging programs from disk
TI Tech Note (T)
A full-screen editor



Issue 5.5

Memoranda Processing: Computer-Assisted Correspondence Made Easy * NanoProcessor: Explore your computer's language * Electronic Typewriter: Make your computer act like an electronic typewriter * TI Card-Trix: Deal out organization on 3x5 file cards * The Plains of Salisbury: Help King Arthur save Camelot * Vital Signs: Control cardiovascular functions with this simulation * Apple Seedlings: Blend frequencies with your Apple * Commodore Hornblower: Create special SID sound effects * IBMpressions: Explore 3-D shadow graphics * Atari Atrium: Make your own "big" sounds with this sound-on-sound recorder * Razzle Dazzle: Crunch numbers down to the 10th decimal * Algorithm-A-Tricks: Learn the mathematical secrets behind Vital Signs * Soundbytes: Merge music and electronics * Product Reviews * HCM One-Liners * Home Computer Industry Journal * Product News * HCM TECH NOTES: Apple, Atari, C-64, IBM, and 99/4A, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

NanoProcessor (A, R, C, I, T)
Learning your computer's language
Electronic Typewriter (A, R, C, I)
The computer typewriter
TI Card-Trix (T)
Organize with file cards
The Plains of Salisbury (A, R, C, I, T)
Battling for Camelot
Vital Signs (A, R, C, I, T)
A "hearty" lesson
Apple Seedlings (A)
Blending frequencies
Commodore Hornblower (C)
Sound effects on SID
IBMpressions (I)
Shadowing 3-D graphics
Atari Atrium (R)
Recording sound-on-sound
Razzle Dazzle (T)
Calculate quick and easy

Atari users please note that coverage in HCM didn't commence until this issue. Also commencing with this issue, programs for the IBM PC/PCjr run as is on the Tandy 1000.

HCM BACK ISSUES



The Plain & Simple Truth About HOME COMPUTER™ magazine

Chock Full of Valuable Software & How-To Articles Without Filler



Every issue is a software "horn of plenty" with dozens of type-in-and-RUN programs printed in an easy-to-read listings format. Our programs are also available on inexpensive disks or cassettes for those who prefer the convenience of ready-to-RUN software. Step-by-step tutorials round out each issue, providing the solid facts you need without fluff or filler. Thus, each issue functions as an excellent reference work, as well as a valuable software source.



No Outside Advertising



Freed from the pressures of servicing *advertisers*, we concentrate on serving our *readers*. Each issue provides uninterrupted editorial flow and graphic layouts for better comprehension—plus unbiased product reviews which focus on true strengths and weaknesses, wherever the chips may fall . . . And we don't have to worry about losing advertisers because of publishing software in the magazine that is "too good." Consequently, we can provide the best free software available anywhere.

Focused on the 5 Hot Home Brands



We are 5 system-specific magazines under one wrapper—not a sprawling, "general interest" publication which attempts to cover too wide a field, only to spread itself too thin. The other side of the coin to this focused approach is the knowledge you gain from being exposed to the many tips, ideas, and techniques we provide for 4 of the 5 systems you may not even have. You'll learn more about your Apple, Atari, Commodore, IBM, or Texas Instruments home computer from this one magazine than from a host of more limited sources.

A Balanced Mix For a Perfect Recipe



In each issue we strive for a perfect balance of productivity, entertainment, education, utilities, and computer literacy—serving the needs of novice and pro alike. Every issue is a full-course meal, with a smorgasboard of tasty dishes for all palates. Whereas other computer magazines may dish out lumps of "editorial indigestion," we serve up a satisfying blend—one digestible byte at a time.

—Welcome to Our World of Home Computing

HOME COMPUTER magazine

Home Computer Magazine (ISSN 0747-055X) is published ten times per year by Emerald Valley Publishing Co., P.O. Box 70288, Eugene, OR 97401. The editorial office is located at 1500 Valley River Drive, Suite 250, Eugene, OR 97401 (Tel. 503-485-8796). Subscription rates in U.S. and its possessions are $25 for one year, $45 for two years, and $63 for three years. In Canada add $7 per year. Other foreign countries $43 for one year surface mail. Inquire for air delivery. Single copy price in U.S. and its possessions is $3.50, and $4.50 in Canada. Foreign subscription payment should be in United States funds drawn on a U.S. bank. Second-class postage paid at Eugene, OR 97401, and Columbia, MO 65201.

POSTMASTER: Send all address changes to **Home Computer Magazine**, P. O. Box 70288, Eugene, OR 97401. Subscribers should send all correspondence about subscriptions to above address.

Address all editorial correspondence to the Editor at **Home Computer Magazine**, 1500 Valley River Drive, Suite 250, Eugene, OR 97401. Unacceptable manuscripts will be returned if accompanied by sufficient first class postage and self-addressed envelope. Not responsible for lost manuscripts, photos, or program media. Opinions expressed by the authors are not necessarily those of **Home Computer Magazine**. All mail directed to the Editor or to the "Letters to the Editor" column will be treated as unconditionally assigned for publication, copyright purposes, and use in any other publication or brochure, and are subject to **Home Computer Magazine's** unrestricted right to edit and comment. **Home Computer Magazine** assumes no liability for errors in articles, programs, or advertisements. Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsement of the product or products by **Home Computer Magazine** or the publisher unless explicitly stated.

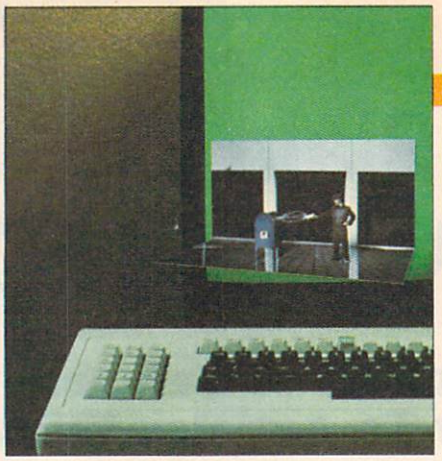
Each separate contribution to this November 1985 issue and the issue as a collective work is Copyright © 1985 by Emerald Valley Publishing Co. All rights reserved. Copying done for other than personal or internal reference use without the permission of Emerald Valley Publishing Co. is prohibited. Requests for special permission or bulk orders should be addressed to the publisher.

Limited License for use of programs in Home Computer Magazine. Emerald Valley Publishing Co. (EVP) is the owner of all rights to the computer programs and software published in this magazine. To allow for use of the software by the purchaser of the magazine, EVP grants to such purchaser only, the limited license to enter these programs into the purchaser's computer, and to place such programs on a diskette or cassette for the purchaser's personal use.

Any other use, distribution, sale, or copying of these computer programs without the written consent of EVP is expressly prohibited and in violation of this limited license and the copyright laws.

Home Computer Magazine, HCM, Home Computer Industry Journal, and Music & Electronics are trademarks of Emerald Valley Publishing Co.

Publisher/Editor-in-Chief Gary M. Kaplan
Executive Editor David G. Brader
Managing Editor Walter Hego
Associate Editor Wayne Koberstein
Sr. Technical Editors
William K. Balthrop, Roger Wood
Technical Editors
D. Donaldson, Tom Green, Edward McManus, G.R. Michaels, Patricia Swift, Randy Thompson, Steve Weersing, Andy Widders-Ellis, Scott Williams
User Group Editor Judy Campbell
Assistant Editors
Laile L. Di Silvestro, Michael Hosmar, Steven P. Nelson, Robin Ogata
Program Translators
Stephan Brittain, Stephen A. Cordon, Jeff Harper, Galen Huntington, James Jentile, Ross Johnson, Robert Paschelke, Francis Saffell
Production Manager Norman Winney, Jr.
Creative Director Gei-Lei Gom
Photography
Nelson Stevens, K.D. Wainsworth
Customer Service **Tel. (503) 341-1029**
Dealer Sales & Distribution **Tel. (503) 341-1036**
Main Switchboard **Tel. (503) 485-8796**



Outside HCM

Zappp! With the crack of a lightning bolt, the power of electronic mailing comes to the home computer. And to the home user, this new power may come as a pleasant surprise—like the pop-up greeting card depicted on our cover. Especially in the Holiday season, most of your correspondence may be as personal as this card. But when your mailing list grows, your computer can apply the speed of electricity to turn that lengthy list into a whole stack of personalized greetings, ready for the mailbox. Such useful software is the "hallmark" of *Home Computer Magazine*—so, look to us. We deliver the mail!

Inside HCM

As electronic tools have accustomed us to quick—if not instant—transfer of information nearly anywhere on the globe, we have come to *expect* this magical ability as part of everyday life. Meanwhile computers have helped speed every level of communications—from the latest video transfer, right on down to one of the earliest forms of civilized correspondence: the mail.

In this issue, we present the *Electronic Postman* himself—a software enhancement to last issue's *Electronic Typewriter*. This modern hero will strike through all the tedium of personal "mass mailing" (just in time for the Holidays!) and still convey your personal touch.

To communicate your personal *power*, take charge of *Serf City*, an economic strategy game set in medieval times. This two-player simulation—pitting one feudal kingdom against another in a race for greater prosperity—will surely test your Machiavellian skills. (Remember: In *Serf City*, "hang ten" means to hang ten serfs!)

Communication is basic to life itself—as you can see in *Cell Mates*, a software simulation of a dynamic living cell. Here, you take the part of the nucleus, communicating life-preserving messages to our internal symbiotic partners: the organelles. As you learn to direct this cellular dance, its very complexity may surprise and awaken you to another great mystery of life.

To communicate a complex idea is one thing—to make it simple enough to understand is another. For, as you will be able to learn from this issue's *NanoAssembler* (a companion to *NanoProcessor* of last issue), simplicity underlies all complexity. Using a simple model of a computer's workings to simulate its complex *works*, this second *Nano* article shows how to use a human-like "assembly" language to communicate in terms the machine can understand.

After communicating our main *HCM* "software instructions," we dedicate specific program/tutorial columns to each computer brand. In this issue, *Apple Seedlings* creates three-dimensional waves on a simulated pond's surface; *Atari Atrium* breeds sine-wave hybrids; *Commodore Hornblower* addresses SID in BASID; *IBMpressions* grows a binary forest; *Razzle Dazzle* evokes some startling sound effects from the 99/4A. And, for all machines, *Problems In Productivity* brings one of our star programs from the past into the arena of everyday life.

In our review section, we try to communicate the real truth about specific products in the computer marketplace: How much hassle is it to add more memory? How speedy is this speed-reading course? Can your computer help you cook up simple financial advice? Can software programs teach you the secrets of the earth? Or will they help you reach for the stars?

Electronic communication may seem so commonplace to us that it takes something special to reveal the magic of electronic media—something as special as music. So as you enjoy our regular editorial fare, take time also to explore this issue's special inserts: the *Cornucopia Sound Sampler* and *Cornucopia Catalog* of electronic musical instruments. We think this is sound advice.

In the Electronic Age, the most powerful instrument—for any kind of communication—is your computer. And at *Home Computer Magazine*, we always aim to keep your home machines well in tune.

Until next time, have fun reading, learning, and RUNing HCM

On Screen

By Gary M. Kaplan
Publisher & Editor-in-Chief

"I've been discussing the part computers are starting to play in personal music creation and performance —now, I get to literally put my music where my mouth is."



Good news. We have a special audio-visual surprise for you. As you know, over the last couple of issues I've been discussing the part computers are starting to play in personal music creation and performance—now, I get to literally "put my music where my mouth is." As narrator of the special bound-in Cornucopia Sound Sampler, I'm proud to introduce the exciting new field of computer-orchestrated electronic music into your livingroom. And to enhance this educational experience, we've prepared an informative Cornucopia Catalog of musical product offerings. This 16-page supplement and our 6-minute stereo recording are the first steps in providing the crucial information that will help home computer users like yourselves to participate in this new electronic music making.*

For best results when playing this stereo recording, I suggest that you first place a regular LP record on your turntable platter, and then lay our Soundsheet on top of it. May I also suggest that you read this issue's *Soundbytes* column; it should help put the relationship between home computer usage and music participation into better perspective.

Once you've played our Soundsheet on the best stereo system you can muster up, please audition the recording for whomever you think may be interested. Don't forget to show them the catalog too.

Anyone entering the world of musical electronics will need a source of timely, accurate, and reliable information. *Music & Electronics*, our new sister publication, is that indispensable source. We're now in the midst of a major drive to gain subscribers to this unique magazine, and we need your help. Of course, we're also hoping that many more of our present *HCM* readers will choose to join us on this exciting new journey.

Producing and distributing a recording and catalog like this takes a great deal of money. We'd like to think that it's worth it. So, won't you please drop me a line telling how you like (or dislike) our efforts. If you'd like to see more of these special inserts, and have ideas for other state-of-the-art products you'd like to see carried, please let me know. Producing a publication like *Home Computer Magazine* without revenue from outside advertising isn't an easy task. If our readership does, in fact, support our Cornucopia product offerings, it makes it that much easier for us to increase the quality of this magazine's content.

On another score (pun partly intended), this issue features some special enhancement software and the premiere of our *Problems In Productivity* column. I'm especially pleased to have had the opportunity to participate in the creation of such a unique, challenging—and badly needed—magazine feature. This new column is a direct result of the reader feedback I asked you for a few issues ago. This kind of exchange is what the reader-publisher relationship is—or should be—all about. And I, for one, would like to see a lot more of it. So please keep your feedback mail coming in. It's crucial in the process of "tuning" *HCM* to your changing needs.

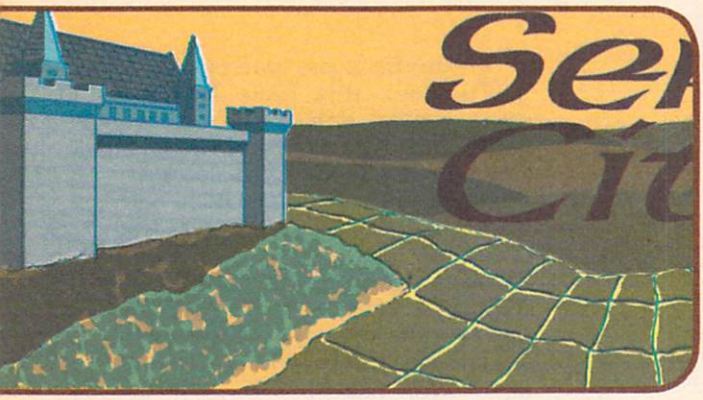
You will notice that in this issue, two of our featured programs are "software enhancers"—separate utilities that add valuable functions to our previous stand-alone programs, turning them into even more powerful, versatile packages. *Electronic Postman* and *NanoAssembler* are important milestones in our continuing quest to bring you the best possible software within the constraints imposed by magazine-delivery economics. If you like this approach, be sure to let us know as soon as possible—quite a bit of lead-time is necessary to design and implement multi-issue software packages.

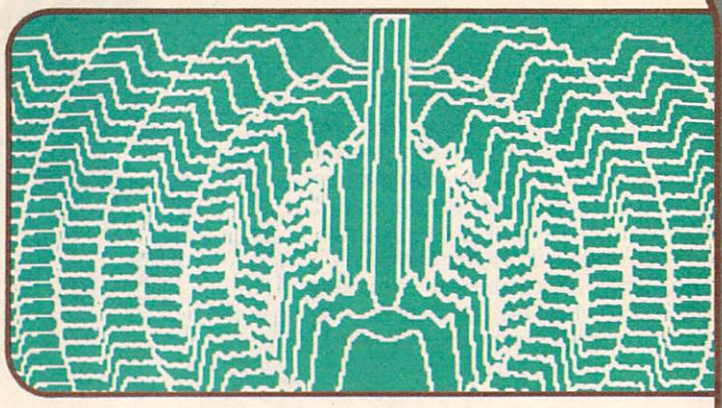
Although the reported marketing machinations surrounding Atari's new 16-bit 520 ST system continue to perplex us, we are nevertheless pleased over the preliminary response to last issue's expansion into the Atari 8-bit world—the 800, 800XL, and 130XE. It's evident from early feedback that *HCM* has a lot to offer the Atari user—valuable material not available elsewhere for so reasonable a cost. It's also obvious that, so far, we've only just scratched the tip of the iceberg in making the large Atari user base aware of *HCM*'s new coverage. I hope that many of you will take the initiative to help us get the word out. And while you're at it, please don't forget to tell all your friends with IBM-compatibles (such as Tandy, Zenith, etc.) and Apple-compatibles (Franklin) that *HCM* now also serves them too. Because we don't have the "big bucks" for expensive ad campaigns, our subscription marketing effort depends on your help. We're here today because of your past support. So, please continue to lend us a hand—come grow with us.



***NOTE TO HCM READERS OUTSIDE THE U.S.**
Because we are presently unable to ship product orders from outside of the continental U.S., the Cornucopia Catalog is not included in non-U.S. distribution. However, Canadian readers will receive the bound-in stereo recording to provide them with a "sound" reason for subscribing to the audio-visual version (magazine plus companion audio cassette) of Music & Electronics.

HOME COMPUTER magazine





FEATURES

15 TI Card Shuffler™
Shuffling files for new Card Trix.
by Randy Thompson, HCM Staff

16 Electronic Postman™
Zap to efficient mailing.
by Randy Thompson, HCM Staff

19 Serf City™
Grow prosperous and rule the valley.
by William K. Balthrop, HCM Staff

22 Cell Mates™
How does a living cell survive?
by William K. Balthrop and Wayne Koberstein, HCM Staff

25 NanoAssembler™
Translating from human to machine language.
by Roger Wood, HCM Staff

44 Atari Atrium™
Blending sine waves on the computer screen.
by Steve Weersing, HCM Staff

46 Commodore Hornblower™
SID sounds even better in BASIC!
by Randy Thompson, HCM Staff

50 IBMpressions™
Binary branches to a recursive tree.
by William K. Balthrop, HCM Staff

52 Apple Seedlings™
A computerized raindrop strikes the pond.
by Steve Weersing, HCM Staff

54 Razzle Dazzle™
A bag of tricky sound effects.
by Scott Williams, HCM Staff

58 Problems in Productivity™
HCM software in the everyday world.
by the HCM Staff

60 Algorithm-A-Tricks™
Highlights of this issue's outstanding software procedure.
by the HCM Staff

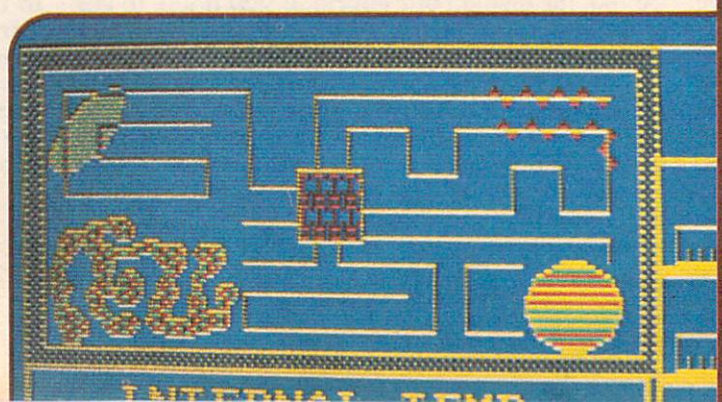
64 Soundbytes™
In harmony with computers.
by Andy Widders-Ellis, HCM Staff

PRODUCT REVIEWS

30 Junior Enhancements
Junior's memory mushrooms.
A Review

33 File Directory
A simple approach to address files and labels.
A Review

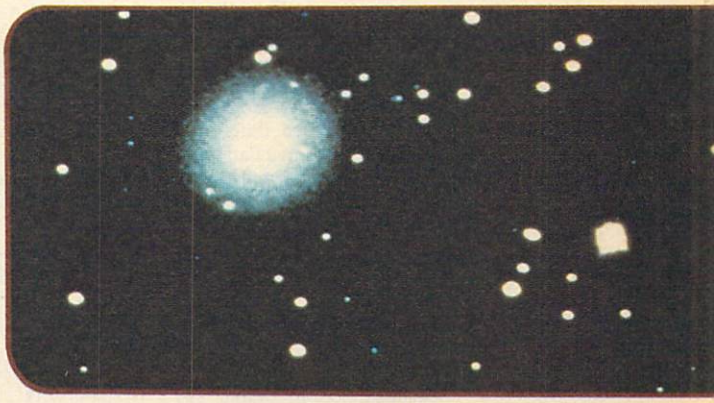
34 Financial Cookbook
Recipes to simplify financial decisions.
A Review



CONTENTS

VOLUME 5 NUMBER 6





36 Earth Science Series — *A Review*
Earth in a nutshell.

38 Reaching for the Stars
A Review of 4 Astronomy Programs — *A Review*
Tour the galaxy!

41 Speed Reading — *A Review*
Speedy lessons—will they help?

DEPARTMENTS

5	For New Readers	73	HCM Bug-Out
6	Outside/Inside HCM	96	HCM Glossary
7	On Screen	—	DeBugs on Display (No Debugs This Issue)
10	Letters to the Editor	Home Computer Tech Notes:	
29	HCM Review Criteria	42	Apple
61	HCM One-Liners	43	Atari
62	Home Computer Industry Journal™	48	Commodore
67	HCM Product News	56	IBM
71	Program Listing Contents	57	TI
71	Programmer's Window Contents		
72	Program Type-in Guide		

SPECIAL End-Of-Year
Warehouse Clearance Sale
On Selected Back Issues of HCM
SEE INSIDE FRONT COVER
LIMITED-TIME OFFER
ORDER TODAY!
Unequaled Value—Don't Miss Out!

MACHINES SUPPORTED

Machine	Requirements	Media
APPLE II family	At least 64K RAM & Applesoft BASIC	ON DISK
FRANKLIN ACE	At least 64K RAM	ON DISK
ATARI 800, 800XL, 130XE	-None-	ON DISK/ON TAPE
COMMODORE 64	-None-	ON DISK/ON TAPE
COMMODORE 128	Must be in 64 mode	ON DISK/ON TAPE
IBM PC	BASICA	ON DISK
IBM PCjr	Cartridge BASIC	ON DISK
TANDY 1000	GW-BASIC Version 2.02	ON DISK
TI-99/4A	TI BASIC or Extended BASIC	ON DISK/ON TAPE

SPECIAL NOTES

Apple Owners: Apple ON DISK media for *HCM*, Vol. 4, No. 1-Vol. 4, No. 3 is in DOS 3.3 format only. Beginning with *HCM*, Vol. 4, No. 4, all Apple programs are in ProDOS format. All programs **RUN** on Apple II+, Apple *IIe*, or Apple *IIc* computers.

Franklin Owners: Beginning with *HCM*, Vol. 4, No. 4, all Apple ON DISK media is in ProDOS format only. Booting ProDOS on a Franklin requires the following steps:

1. Boot ProDOS. When the system hangs up, press [RESET].
2. Type 265B:EA EA and press [RETURN].
3. Type 2000G (insert no spaces between the last zero and the G) and press [RETURN].

See *HCM*, Vol. 5, No. 4, page 13 for more information.

Tandy 1000 Owners: Starting with *HCM*, Vol. 5, No. 5, all of our IBM PC programs run on the Tandy 1000 without modifications. Programs prior to Vol. 5, No. 5 may need minor changes as explained on page 130 of *HCM*, Vol. 5, No. 4.

Errata

Atari Computer Users: Due to an oversight in our software that translates program format to typeset listings, line 390 of the *NanoProcessor* in *HCM* Vol. 5, No. 5 mistakenly contained an English Pound symbol (£). This character should have been a backslash (\), which you produce by pressing [SHIFT] + on the Atari computer's keyboard.

Attention Subscribers:

To avoid missing any valuable issues, please check the address label for maturity of your subscription—so you can be sure to renew before it expires.

Letters to the Editor

Atari Saves A Reader
Dear Sir:

Today I received your magazine and much to my surprise, I found that you have added Atari to the micros covered. I must tell you that I was an old TI subscriber who had upgraded to the Atari and though I found you a source of good info and programs when I had the TI, I was going to let my subscription lapse—not now!

Enclosed please find a check for $5.95 for the Atari disk with the programs from Vol. 5, No. 5. If you can produce the great data, info, and programs that I know you have had in the past, not only will I renew, but I will talk to some of my friends and show them what you are all about.

Keep up the good work, and I and a million (or more) other Atari owners will flock to buy your magazine. I especially am fond of your tutorials and short programs which are usually both instructive and entertaining.

Alex Pignato
Oceanside, Long Island, NY

Alex, your letter is typical of the positive response we have received to our new Atari coverage. We are already beginning to see the vanguard of that flock you speak about. We want all the Atari users out there to know that the roost is ready at Home Computer Magazine. So, fly on over!

Better Sequential Saver
Dear Sir:

I am a proud owner of a Commodore 64 and enjoy HCM very much. In last month's issue, Mike Poole explained how he used the C-64 merge program to convert sequential files to a usable program. I think I have come across an easier way to make a sequential file a usable program. Just type LOAD"PROGRAM NAME,S",8. The sequential file is loaded into memory and can be saved as a program file.

You can also save a program as a sequential file by putting (,S) at the end of the file name when saving a program. This serves as a form of protection from those who don't know about this trick. If you put (,U) at the end of a file name when saving a program, the program will be listed as a user file in the directory. The program can then only be loaded with the ,8,1 ending. I hope this information is useful to some HCM users because it has been very useful to me.

Bert Palmer
Address unknown

Well Bert, saving a program with the ,S extension does not actually change the format of the file—it is still a program file. This method does, however, place the SEQ identifier next to the file. Thus, even though the directory contains the SEQ, it is possible to load a program file saved in this manner because the file is still in program format. Your method does not work with program files that were listed to disk as sequential text files (see "HCM Bug-Out" for details on how this is done), and therefore not in tokenized format. The program must be in tokenized format for the LOAD statement to work properly. As for program protection, using the extensions ,S and ,U is a great little trick. Who would think that a "SEQuential file" was actually your latest programming masterpiece? And better yet, who would think to LOAD it? Thanks for the tip Bert.

Creative Cursoring On Apple II
Dear Sir:

Did you ever wish that you could change the Apple's cursor without any special routines? Well, I have. On an Apple, screen memory is larger than the actual screen. I guess it was easier to just make it an even 1-K. The leftover bytes are either unused or are used for special functions. Location 2043 (decimal) is the value of the cursor (its ASCII value + 128). If you POKE this location with the correct number, it changes the cursor to whatever you want. For example,

```
POKE 2043,ASC("_")+128
```

will produce a blinking underline cursor. This cursor remains until you POKE 2043 with a different number or press [CONTROL] [RESET]. If you put the actual ASCII value here, the character will be in a flash mode in addition to the normal cursor blink. Am I the only one who didn't know this, or is this an important find?

Donald W. Scott, Jr.
Address unknown

Donald, you've found something that we haven't seen documented before—but don't try to get it to work on an Apple without 80-column capability. The location you are POKEing is one that is part of text-screen scratch-pad RAM used by expansion slot 3 or the auxiliary slot on the Apple IIe. Apparently, this location is used by the 80-column card to determine the cursor character, even when the computer is not in 80-column mode, which explains the little trick you've uncovered.

Who Adds Most To PCjr?
Dear Sir:

As an owner of the PCjr, I have several questions that are in urgent need of reliable answers.

1. There are five sources (Tecmar, Quadram, Impulse, Racore, Legacy) of second-drive enhancements with memory expansion for the PCjr, but what are the special features of each? How does each of them increase the PCjr's compatibility?

2. Can the PCjr utilize the 8087 math-coprocessor? (For example, the Micro Way 8087 math-coprocessor for PCjr.)

3. Is there any other internal modem besides the one sold by IBM that the PCjr can use?

4. How is Racore's DMA controller board utilized by the PCjr? Does it make the PCjr more compatible?

5. Because of the PCjr's lack of DMA, intense I/O and downloading of large files are close to impossible. Would the DMA controller board by Racore eliminate such problems? Also would it allow the disk drive and the CPU to work simultaneously?

Wen J. Chen
Flushing, NY 11355

Over several issues, we have reviewed the Tecmar jrCaptain (Vol. 4, No. 4), Legacy II (Vol. 5, No. 2), the Quadram Expansion (Vol. 5, No. 4)—and, in this issue, both the Microsoft and IBM PCjr memory expansion systems. In this last article, you will find a chart comparing the features of all these products. We are currently examining the Racore version of the previously reviewed Quadram product and will publish a review in an upcoming issue. We are also looking at the Impulse product and the Racore DMA controller as future review items. In addition, our technical staff is researching the feasibility of installing an 8087 in the PCjr—look for an HCM report in an upcoming issue.

TI LOGO II Cartridge Sought
Dear Sir:

I would like to buy the TI LOGO II cartridge. I have been looking for a place to buy the cartridge, but cannot find one. Could you help me out? I would greatly appreciate it. Thank you.

Roman L. Schmitz
Malone, WI 53049

Roman, we've traced down a few sources for you. TI LOGO II is currently available from: TexComp (P.O. Box 33084, Granada Hills, CA 91344) for $20.00; Unisource (P.O. Box 64240, Lubbock, TX 79464) for $69.95; and Triton Products (P.O. Box 8123, San Francisco, CA 94128) for $19.95. Other local or regional sources may also carry this product.

Disenabling Break Key?
Dear Sir:

To disenable the break key on the IBM computer, this program works:

```
10 POKE 108,64:POKE 10,1:POKE 110,112:POKE 111,0
```

I tried this program on the Tandy 1000—it did not work. Another way to trap the break button on the IBM is this program:

```
10 KEY 17, CHR$(4)+CHR$(7):KEY (17) ON:ON KEY (17) GOSUB 100
20 REM REST OF PROGRAM HERE
90 REM
100 PRINT "BREAK BUTTON INOPERABLE":RETURN
```

This did not work on the Tandy 1000 either. I hope you can solve my problem. Thank you.

Darren Eichorn
Hatboro, PA 19040

Darren, the BREAK key on the IBM is function key 17, and is therefore redefinable as you have detailed. The Tandy machine does not allow redefining of the function keys above 12. Consequently, no similar fix that we know of will work on that machine. Are there any Tandy programmers out there who've been able to perform this feat of magic?

Program Prompts Purchase
Dear Sir:

I bought a computer simply because of your program "The Organizer." I had wanted "Think Tank," but knew it would only run on Apple or IBM PC machines and, in addition to its price (approximately $125), I felt the overall system costs of these two machines were above my means at this time; but when I saw the "The Organizer" would run on the Commodore 64, I then bought a second-hand 64 (and your software)! I have not yet tried it out, so my enthusiasm may be premature, but at least I've gotten my feet wet.

I would like more technical things and fewer games. Also separate loose-leaf documentation that we could order from the magazine, for ease of use and safety (cheap of course). I appreciate the documentation in the magazine, but the magazine will be hard to use everyday and I will have to make a photocopy.

I notice there seem to be quite a few programs exclusively for the Apple or IBMs. I hope Commodore doesn't discontinue the 64, I would like to see more 64 programs in your magazine.

The program listings in your magazine are helpful even if I don't key them in. If you print updates on the programs I use, I will have to be able to find the lines easily.

Thanks for innovation!

Lin Sinclair
Alameda, CA 94501

Thank you, Lin, for the compliments and advice. Actually, if you look more closely, you should find that our magazine content is evenly balanced between all the machines we cover. You can check this out yourself by counting through the Program Listings Contents on page 71. As for technical material, almost everything we publish contains a wealth of technical information—e.g., our Programmer's Windows for all major programs (including games), plus Tech Notes, tutorials, and reviews of specialized products. We try to balance this material against lighter content for less tech-oriented computer users. A word about the "games" we do publish: We have recently shifted our approach to computer gaming away from the arcade to simulations with some educational—as well as entertainment—value. Your suggestion of providing separate software documentation is interesting and deserves consideration. Anyone else out there want to see this? Good luck with your new computer, Lin, and have fun!

MousePaint: Look But Don't Print?

Dear Sir:

I think your magazine is marvelous. We just recently purchased an Apple *IIc* computer and have bought two of your Home Computer Magazines (Vol. 5, No. 2 and Vol. 5, No. 3).

My question to you has to do with my printer and my mouse. It seems I can't get the MousePaint pictures I drew to print on my Star Gemini 10x printer with the PAC interface. We've tried everything and hope you can shed some light on the subject. Will you please explain what might be wrong? Is the problem in my printer, interface, or MousePaint disk?

I was wondering if you were planning on putting more arcade-style programs in your magazine in the future. I also think you should arrange your magazine into sections, like putting all the Apple software in one section, C-64 software in another section, and so on. This would really help me and probably all the other subscribers a lot.

Roger Thompson
Reeds Spring, MO 65737

In order to send a MousePaint drawing to a printer, you need an interface and printer combination that will dump the high-resolution graphics screen. From what you say, it doesn't sound like your interface-printer combination can do this. We've had excellent success using the Grappler+ from Orange Micro (1400 N. Lakeview Avenue, Anaheim, CA 92807) and an Epson MX-80. (See "Mousing Around on the Apple II" in HCM Vol. 4, No. 4 for details.) As to your question about separating the magazine into sections: If you look at the issues following Vol. 5, No. 3, you will see that we have indeed phased in a system of marking the magazine in easy-to-identify content sections. Special "edge-tab" markers flag each major section. Documentation for all major programs (those with versions for all 5 computer brands we cover) appears in the first section marked "Software Instructions." This is followed by a section of "Product Reviews," then "Tutorials" (including separate columns for each computer brand), the "Programmer's Windows," and finally the "Type-in Listings" section. We hope this will be a very handy quick-reference guide for you and others who have requested similar tab-indexing.

To DeBug, Or Not To DeBug . . .

Dear Sir:

As a new subscriber to HCM I wish to tell you how pleased I am with your magazine. I purchased the first issue I ever saw at a magazine stand and was surprised at the amount of information I was able to understand and use. I am a newcomer to computers and have a TI-99/4A computer with a TI Expansion system, 32K memory board, RS232 interface and Extended BASIC. I am still not sure of how to make the most of this system, but am enjoying every moment at the keyboard.

After going through Vol. 5, No. 3, I called your 800-number and ordered a one-year subscription, and backorders of Vol. 5, Nos. 1 and 2 ON DISK. They have just arrived and after reading instructions on how to update the disks (I am referring to The Organizer program) and a little trepidation, I began. Then to my utter surprise, I discovered that the updates had already been done to Vol. 5, No. 1 ON DISK. Did I somehow miss some information as to these updates being made automatically, and if so, can you explain the procedures for when and if updates are needed?

I realize that Texas Instruments is no longer making the TI-99/4A, but I am sure that there are many other new users who might appreciate some articles that are geared for beginner users. I am referring to some of the lesser-used functions such as PEEK, INIT, LINK, DEF, etc. When do you use them, and examples in short programs? How about a LOAD program to automatically boot your disks?

Thanks for a terrific magazine which is giving me hours of pleasure.

Sylvia Kesten
Northridge, CA 91326

Glad to have you on board, Sylvia. After checking the introduction to our DeBugs section, we can understand your confusion. Sure enough, we do not specifically communicate this fact: ON DISK issues always contain the latest debugged—and enhanced—versions of each program as they are shipped. In addition, the most recent ON DISK issue will also contain an "update file" of the latest DeBugs, for those who received any previous issue before all its DeBugs appeared in print. These files can be merged with programs on the older, non-updated disk. If you want to ascertain whether your disk contains the latest updated programs or whether it requires merging in update files, compare the program's version number on your ON DISK issue to the version number of the latest DeBug as it appears in print. If the numbers are the same, your version contains all the latest DeBugs; you do not have to merge it with an update file.

We have—and will continue to—run articles that demonstrate specific "PEEK and POKE" kinds of programming methods. For the 99/4A, check out "Razzle Dazzle," a regular tutorial column that beginners can learn from.

Putting A Second Drive Into Junior

Dear Sir:

As a loyal reader of HCM for several years, I am pleased to see that you are still keeping readers up-to-date on happenings with the PCjr. As you probably know, most other magazines concerning the PCjr discontinued publication when production came to a halt. All PCjr material which I have read in HCM is always very helpful and beneficial to me.

Now I need special help! Shortly after I purchased my PCjr, I bought an Okidata printer. Next, I purchased the Tecmar jrCaptain. With these new additions, I have been able to utilize my PCjr for something other than playing games. My problem is this: I would like to purchase a second disk drive. I already own a clock/calendar, a printer port, and have expanded memory to 640K. I have shopped around, looking at the Legacy and other systems, and am very disappointed because I don't want to pay hundreds of dollars for a second drive which adds features that I already own! All I want is a second disk drive—nothing else!

Please help me locate a second drive for my Junior, where I don't have to pay for unnecessary features.

Stephanie Fox
Brick, NJ 08723

At HCM we have several PCjr's with second disk drives, Stephanie—and they are completely compatible with the Tecmar jrCaptain. (For specifics on how a second drive and the Tecmar product work together, see the letter from Ron Sutherland in HCM Vol. 5, No., 3.) We installed these drives ourselves using the method described in the article entitled "One For The Money, Two For The Slow: Adding A Second Disk Drive To The PCjr" in HCM Vol. 4, No. 4. You may wish to get a copy of this issue (see HCM Back Issues in the front of this magazine), and see if this solution fits your needs. The kit described in the article allows you to modify your PCjr disk controller card to support a second drive, and is available from our order department for $49.95 plus $3.50 for shipping and handling.

Seeking C-64 Interface

Dear Sir:

I have read your fine magazine since the Vol. 4, No. 1 issue, especially for the Commodore 64 articles and the Letters to the Editor column. This question is directed to that column: What interface(s) will work best with the Commodore 64 and the Gemini-10 (not the new 10x)? One that would give complete Commodore 1525 emulation?

After many unanswered letters from printer interface companies, I'm hoping the answer to this question shows up in your Letters to the Editor column. Thanking you in advance.

Richard L. Wightman
Wellsville, NY 14895

There are so many good interfaces, Richard, we don't know where to begin. Cardco (300 S. Topeka, Wichita, KS 67202) has several quality interfaces that make the Gemini-10 emulate a 1525. If all you do is word processing, the ?/B or ?/PS Cardco interfaces should suit you fine. If you want something with graphic capabilities, the ?/G+ from Cardco is an excellent choice. Both The Connection from Tymac Controls Corporation (127 Main Street, Franklin, NJ 07416) and Super Graphix from Xetec (3010 Arnaold Rd. Salina, KS 67401) provide 1525 emulation with graphics.

Making Room For Mac

Dear Sir:

I have noticed that your last two issues of Home Computer Magazine have had a section called "MacRos" for the Apple Macintosh. After seeing these articles I have two basic questions for you. First, are you going to continue to support the Macintosh? Will you broaden the coverage? I hope you will, especially since the Macintosh is a member of one of the families you are supporting. (When IBM introduced its PCjr, you immediately started to support it!)

The second question deals with the MS-BASIC for the Macintosh. If you start to support the Mac, will you provide programs for it? I know that the MS-BASIC version for the Mac is similar to the MS-BASIC for IBM. If so, it would seem to be easy enough, even without color. Please consider what I have suggested. I really am interested in buying a Mac and I would like your magazine to cover it in the same style which you have covered the other computers. I am sure that other readers share my view—so please consider it.

Brian Neidig
Taylor, TX 76578

Brian, we have run several features for the Mac in recent issues to see what kind of response we would get from home users. We have to say that the response—though positive—was sparse. Your letter is one of the very few we have received from Mac users. This is not to say that—given more requests—we won't cover the Macintosh in the future. But to include Mac versions of all our major programs, we would need to make space in the magazine by deleting something else or increasing our total page length—a costly alternative in either case. There's also another consideration, Brian. Apple just recently canceled its long-worked-on MacBASIC. Our plea to the Apple for bundlng the languages with all Macs is now moot. At its present price in the $150 range, we feel that Microsoft's BASIC for the Mac is too costly a package for a large enough user base to develop. Without the necessary numbers, an expansion to full Mac coverage just doesn't make economic sense.

99/4A Lock-Up

Dear Sir:

I own a TI-99/4A and have been having problems with the computer locking up. I have been told that it is normal for it to happen once in awhile, but it locks up much more than it should. The cartridge that locks up the most is Extended BASIC. When I plug in the Extended BASIC cartridge and then select choice #2 for Extended BASIC the screen will sometimes fill up with random symbols.

It also locks up with other cartridges. For example, in the middle of a chess game with the computer, the computer will sometimes lock up. This problem appears to be either in the cartridge plug-in or where the peripheral expansion system plugs into the speech synthesizer or possibly where the speech synthesizer plugs into the keyboard.

I have written many letters to different companies and nobody seems to be able to figure out my problem.

Any help you could give me will be appreciated.

Kip A. Dondlinger
Hutchinson, KS 67502

Kip, you have just hit on two of the 99/4A's trouble spots. The cartridge slot and the side port are very susceptible to wear and tear. The side port, in particular, can give users many headaches. To help minimize your own headaches, make sure that the console and expansion box cable are on a flat, level surface. Problems start arising if the expansion cable doesn't fit just right into the side port. The side port can also be a problem if your console moves around on the table top. Any time the console moves, the connection between the console and the expansion cable can weaken. If this is a problem, try putting double-sided tape under the feet of the console to keep it from slipping. Finally, the cartridge port is quite often the culprit in a lock-up. Extensive plugging and unplugging of cartridges would cause even the most expensive connectors to wear out in a very short time. Unfortunately, it would require some technical expertise to disassemble the cartridges and/or your console to clean the contacts or, in severe cases, to replace the contacts. The best solution to this problem is to minimize the number of times that you switch cartridges.

Ask And Ye Shall . . .

Dear Sir:

I don't own an Atari 800XL computer anymore. I sold it because it didn't stand up to my TI. Why don't you cover the Atari computers? There are more Atari 800s and 800XLs than PCjrs, not to mention Atari 400s, 600XLs and some 1200XLs. If you added Atari's line of computers to your magazine you would really see a jump in your subscriptions and sales. If space is a problem why not just drop the PCjr? (I know that will make a lot of people angry and upset, so let me apologize now to any offended person.)

Also you have HCM one-liners. Why not HCM graphics? There are many people who would love to show off their still and moving graphics programs. I feel you should let your faithful readers know the answers to these questions.

Frank DeCandia
Jersey City, NJ 07307

There's no need to drop a very fine machine like the PCjr, Frank. We have kept Junior and also added more pages for Atari coverage. As you can see from Mr. Pignato's letter (Atari Saves A Reader) above, we are already starting to reap the benefits of this new coverage.

In regard to your second question, our "HCM One-Liners" page generally includes outstanding graphics submissions. In addition, our specialty articles – like TI "Razzle Dazzle" and "Atari Atrium"—make perfect showcases for such programs. If you have an original program that you'd like to show off, send it to us. We welcome any article and program submissions—graphics or otherwise.

For IBM PC Sans Graphics Board

Dear Sir:

I recently bought a copy of your magazine at a newsstand thinking I would enjoy running several of the programs offered. I found, however, that I couldn't use any of the programs because my IBM PC is not equipped with a graphics card. I bought the computer for data management (text) and word processing, so a graphics card is unnecessary (and would cost about $500). Wouldn't it be possible with a program like Budgetron to have an alternative version for machines without graphics? If not, then your publication is of no use to someone like me, which seems a pity.

Thank you for your attention.

Katherine A. Kirk
Victoria, BC, Canada

Katherine, the architecture of the IBM PC is unlike other computers such as the Apple. The IBM BASIC interpreter only supports graphics commands when the graphics adapter is installed. We feel that color is an essential part of good home-user software. Because both the IBM PC and PCjr are such great color-graphics machines, we don't want to limit our readers to monochrome-only programs. Without the graphics interface installed, our programs would be limited to text only. Whenever a program runs best with the card, we consider it a system requirement. Budgetron is an excellent example: it will run without the graphic adapter card, however the bar-graph option results in an ILLEGAL FUNCTION CALL. If you don't select this option, the program will work just fine. After running some tests, we have determined that the following programs will run on the IBM PC without a graphics adapter—with limitations as noted.

Vol. 4 , No. 2: Electronic Home Secretary
Vol. 4 , No. 3: Snap-Calc and Missile Math (without space ship graphics)
Vol. 4 , No. 4: Basic Addition and Subtraction and Tax Deduction Filer
Vol 4, No. 5: Division Tutor, Personal Loan Calculator, Quiz-Make, and Quiz-Take
Vol 5, No. 1: Quiz-Print, all Organizer programs
Vol 5, No. 2: FNkey, Organizer Reports, It Figures!
Vol. 5, No 3: Geometrix (with limited graphics), Budgetron (no bar graph option)
Vol. 5, No 4: Run-Day-View, Trig-Trix (with limited graphics)
Vol 5, No. 5: Electronic Typewriter, Bug-Out
Vol 5, No. 6: Electronic Postman

Bug-Out Identifies Bugs
Dear Sir:

I own an Atari 800XL with a single disk drive, a tape recorder, and a very dotty matrix printer. I have previously owned four other computers, all of which were ignored, for the most part, by the publishing industry. It was really nice to see the support for my machine at last.

This reaction is tempered a little after a couple weeks of (intermittently) typing programs from your magazine! The first thing I typed in was the Proofreader, and I must admit that it worked. The second thing I undertook was the NanoProcessor and my experience was not so great.

The first problem arose when I tried to enter line 920, a DATA item with a clear and unmistakable Pounds-Sterling symbol. Atari X models have such a symbol; however, the use of it normally excludes the other graphic characters used in the NanoProcessor display. I eventually figured out that what you must have meant was a **SHIFT +** character used (by Atari mostly in other languages than BASIC) for mod division.

The second problem was rather more of a bug, and took a while to figure out. The program checks the location 764 for last key pressed, and branches accordingly. At least, it ought to; it doesn't. The problem seems to be well illustrated by the segment from line 1500 to 1580. The variable B contains the last-key-pressed identity from location 764; line 1520 tries to match it against the ASCII value of the control-panel keys. But the Ataris use three interlocking character set designations: ASCII; Internal code, which is the sequential position of the character in the shape table; and Keyboard code, which bears little resemblance to the other two. Keyboard code is tied to the row-column position of the keys, and it is a member of this set which is stored at 764. Here's a one-liner for you to illustrate the point:

```
1 FOR A=1 TO 256:A = PEEK(764):
?CHR$(A);'' '';:NEXT A
```

PLEASE type it EXACTLY as shown. The continuous reassignment of the counter variable value within the loop keeps the thing going forever, or until you have had enough (use BREAK to get out). Press any key, alone or with either or both of CONTROL and SHIFT keys. Try the Atari key (inverse toggle) and the CAPS, ESC, and RETURN keys. Try using each key with and then without the CONTROL. Try any key presses you like. CTRL-1 which pauses all action and then unpauses it, and the BREAK key will not give unusual results.

But back to the program bug. I thought of looking for the keys that would give the result, in effect reassigning the control panel in the game. Almost immediately after that, I thought of altering line 1580 so that the correct connection with ASCII would result. Both of these approaches had to be rejected because not all ASCII values represent valid keycodes. Not even all 6- or 7-bit numbers are valid. As a matter of fact, it turns out that some of the values required for the program are not valid K-codes. The only workable solution I found was to rewrite the following two lines.

```
1520 READ ZQ: IF B=ZQ THEN 1550
1580 DATA 49,50,51,52,60,62,80,69,6,
     73,82,76,62
```

I have not yet had a chance to play with the program, to see whether or not it is completely debugged. At least these fixes get it started running.

I look forward to the next few issues of your magazine. I hope it lives up to the high promises made by issue 5.5, apart from these details.

Diane Hoffman
Oshkosh, WI 54901

We're glad to have such a knowledgeable Atari reader, Diane. Thanks for the comments. The first bug you noted was due to an oversight in our process for translating the program to our type-set listing format. The other bug seems to stem from an error you apparently made in typing in the program. Line 1460 does indeed PEEK location 764, but B does not contain the value from that location—it contains the ATASCII returned from the GET #2, B statement. Thus, that aspect of the program works correctly as published. Thanks, though, for the excellent discussion of the various ways Atari machines represent key input, as well as for the neat little "one-liner."

Word Pro From ProDOS To 3.3
Dear Sir:

I am having some problems with my Apple *II*c. A few weeks ago, a friend of mine sent some documents he created with his word processor to me. The documents were on a ProDOS disk and were stored as text files. My problem is that most of the documents he sent to me that were stored as text files were stored under a sub-directory. My word processor can read text files when they are directly stored as text files, but not when they are stored under a sub-directory. My word processor translates the text files to files my word processor can understand. (The files are on a DOS 3.3 disk now because that is the only format my word processor can translate them from. I copied the files from the ProDOS disk. This procedure has always worked until now.) Do you know of any program or procedure that would allow me to store these text files directly onto the disk instead of under a sub-directory so I could print the files on my printer with my word processor? I need a solution really soon.

Also, I have heard about a Z-RAM board by Applied Engineering that would allow you to add 256 or 512K of additional memory and a Z-80 microprocessor to the Apple *II*c. Their ad says that it comes with a RAM disk so programs could run 20 times faster. It also said that you could run regular Apple software and CP/M software without unplugging the board. Do you think you could review this product? It sounds very interesting.

Thank you very much for your help.

Pete Sison
San Diego, CA 92139

In order to convert a file from ProDOS to DOS 3.3 you need two disk drives and the ProDOS Convert utility from Apple. If you have these, the various sub-directories shouldn't give you any trouble in conversion—just specify the complete ProDOS pathname in the convert program. We are evaluating the Z-RAM card from Applied Engineering and will be reviewing it in a future issue.

Hard-Driving Junior?
Dear Sir:

Like thousands of others, I thoroughly enjoy your magazine. In particular, I have used your techniques to successfully add a second floppy drive to my PCjr. Now I wonder about adding a hard drive. With the rapid de-escalation in hard drive prices this would appear to be a valuable next step. Any possibilities of your experts telling us how to do that? I am sure a number of us would benefit greatly.

Warren T. Dent
Indianapolis, IN 46260

Warren, our experts are currently working on hard-disk drive solutions for the PCjr, so stay tuned to HCM for cost-effective means to expand your PCjr's capabilities. Also, if any other readers desire additional Junior enhancements, please let us know so we can work toward giving you what you need.

Terminal Tape Troubles
Dear Sir:

I recently purchased a modem through mail order for my Commodore 64. Somehow there was a mixup in my order and the company I ordered from sent me the terminal software on disk instead of cassette tape as I requested. I contacted the company and they told me that the terminal software was unavailable on tape. When I borrowed a friend's disk drive to load the program and try to transfer it to tape, I found that I could only access one line of the program which was a SYS command. Is there any way I can access the full terminal program and copy it to tape or will I be forced to purchase a disk drive to use my modem?

Matthew Correll
Grinnell, IA 50112

Buying a disk drive is not a bad idea Matthew—especially now that the price on the Commodore 1541 is so low—but it is not the only answer. Several fine terminal programs come on tape and even cartridge. The popular VIP Terminal from Softlaw is available on tape. The SkiWriter II cartridge from Prentice-Hall (Reviewed in HCM Vol. 5 No. 4) is both a word processor and terminal program. As for copying and modifying the program you have now, don't try. It may be copy protected, and the attempt may void its warranty. If what you have does not work with your current system, we suggest that you either buy a disk drive or return the package.

Mini-Amnesia On Mini-Memory
Dear Sir:

The Mini-Memory module for my TI home computer will not accept assembly programs from tape, or BASIC programs from the computer. I assume the battery has died.

Could you tell me if this can be remedied and if so, where do I send it? How much would it cost? How long would it take? Have I diagnosed the problem correctly?

Any information you could send me would be appreciated.

John Wood
Salem, OR 97306

John, it's good that you should bring up this subject. There are a lot of Mini-Memory car

tridges out there with weak or failing batteries. There's a very good possibility that your Mini-Memory battery has finally gone to that great recharger in the sky. The symptoms you describe fit the dead-battery syndrome, though that doesn't rule out the possibility that the connector could be dirty or the cartridge misaligned so that the pins on the card edge don't fit correctly when the cartridge is inserted into the system. Fortunately, there is help on the way. We are currently tracking down the least-expensive sources for the battery, and will tell you how to replace it in next issue's "Home Computer Tech Note" for TI.

Personal Accounts-Payable?

Dear Sir:

I have an IBM PC and I am looking for a computer program that will do the following things. I was wondering if you would have this on any of the disks that are available through you, or if you might know where I could find it.

It is to manage my personal bills. I would like to be able to enter the company name, company address, or a code number for each company, subtract the payment made and have the balance show. The next month I would like to be able to subject the payment again and have the account balance show. It would work like an accounts payable file.

Thank you for your help.

Florence M. Glenn
Santa Ana, CA 92702

Florence, so far we have not published a program that will totally "fill the bill" for you. We are, however, developing such a program—one that will not only allow the kind of file you suggest, but also provide great flexibility in customizing other kinds of alphanumeric/calculating data files. Look for this software soon in an upcoming issue.

Looking For Artificial Intelligence

Dear Sir:

I will be a sophomore in high school next year and have acquired (through certain channels) a free hour each day which I spend working with Apple IIe computers. I also have a Commodore 64 at home and I am planning to buy the new C-128 when it arrives on the market.

My question is this: Do you know of any companies that specialize in literature and/or software on artificial intelligence programming? I intend to focus my future computer studies on this particular subject.

Thank you.

Philip G. Jones
Slater, MO 65349

Philip, your local library will probably have a few major books on Artificial Intelligence (AI), although such books are largely non-technical and are targeted for the popular market. Most famous among these is The Fifth Generation, *by Feigenbaum and McCorduck, which talks about Japan's push for AI. Others include:* Artificial Intelligence, *by L. Stevens,* The Thinking Computer, *by B. Raphael, and* Godel, Escher, Bach, *by D. Hofstader. If you want technical information and news of the current state-of-the-art, a newsletter published out of Scottsdale, Arizona, called "AI Trends," will fill your needs. You will find that most AI research these days, centers on "expert systems"—software that provides the kind of technical advice usually reserved for experts in a given field. For example, Hewlett-Packard employs an expert system on the assembly line to help diagnose defects in the structure of microchips. These programs use a combination of raw data bases and a "knowledge" base garnered by quizzing engineers on every conceivable "what-if" situation. Some of these professional techniques have trickled down to the microcomputer level—with programs like Personal Consultant for the TI-99/4A, Experlisp for the Macintosh, and the Knowledge Engineering System for the IBM PC. Human Edge Software offers a few such "decision making" programs for the Apple II family; but we don't know of any available for the C-64 or the C-128. (The C-128 should be compatible with AI software written for CP/M systems, however.) As these kinds of programs become more applicable to home use, we will certainly cover them in HCM.*

A Pound Sign For A Backslash

Dear Sir:

Imagine my surprise. I have never looked in your magazine, just at the cover. This time—Atari!

Your magazine seems much more imaginative than most of the others, especially in your programs and explanation sections. I see an excellent relationship growing.

However, is it just because you are new to the Atari field? Your NanoProcessor program in Vol. 5, No. 5 has a bug that can't be just a typo. Line 920 has a data statement which includes an English pound sign! I can access this symbol on my 800XL with the International Character function, but it reverts to a simple CONTROL 'H' graphics symbol on LISTing or RUNning. What did you have in mind?

Second, I can't get the program to work. I have CAREfully checked my listing but when I RUN the program I get an Input Statement Error (#8) on line 820. For some reason my computer will not accept the line 920 data statements. If I remove the offending statements (losing the switch pointer) the program runs for awhile until I enter the third line of the sample program where I get a String Length Error (#5) in line 1660.

Last, but not least, Atari Bug-Out doesn't DO anything! It runs, it asks for output device, it says READING, but it doesn't access my cassette file. Is it supposed to operate my cassette recorder? Your instructions don't say!

I know this letter looks hyper-critical, but it's not meant that way. I'm super impressed with your first Atari issue. I know it can only get better.

Norm Palosky
Alexandria, VA 22304

Thanks for the input, Norm, and welcome aboard. For the answer to the first problem you mentioned, see "Bug-Out Identifies Bugs" (above). As for the Bug-Out program, other readers have experienced no difficulty; we can only conjecture that you have made a typing error.

No ProDOS-Compatibles

Dear Sir:

I have three different Apple II + compatible computers (from two different manufacturers) that I have been using heavily for over two years. I haven't had the slightest compatibility problem with loads of different software and with different operating systems, including CP/M and Pascal. But ProDOS doesn't work. None of the ProDOS diskettes that you have sent me works. Neither does one recently purchased program which is on a ProDOS diskette. The disks boot up displaying the title screen for the ProDOS release, then the computer locks up. The only key that the computer will respond to after that is RESET, which causes more problems. (Note: Two of the computers have 16K RAMcards in slot 0, and the other has a 128K Saturn board.) Is it necessary to purchase a ProDOS preboot diskette? Or is there some multiple-letter code to type in? Or has Apple figured out a way to defeat the "compatibles" with this latest operating system?

If I'm out of luck with ProDOS, would it be possible for you to send me your Home Computer ON DISK software with DOS 3.3?

Mike Rosa
Peru, IN 46970

Mike, on the second page of the Table of Contents in this issue, there are instructions for running ProDOS programs on Franklin machines. You don't say which brand of Apple-compatible you have, so we can't guarantee this will work for you. In Vol. 5, No. 5 , a letter from Henry Curry provides another way of fixing a ProDOS disk for Franklin—by using a "Sector Editor."

Expansion Of Coverage Hailed

Dear Sir:

This is the first time I have ever written to any magazine and I just wanted to say thanks. I am an avid Atari user and will help support anyone who supports Ataris. The first time I saw your magazine was several months ago on the newsstand. I was disappointed to see that you did not include Atari in your coverage.

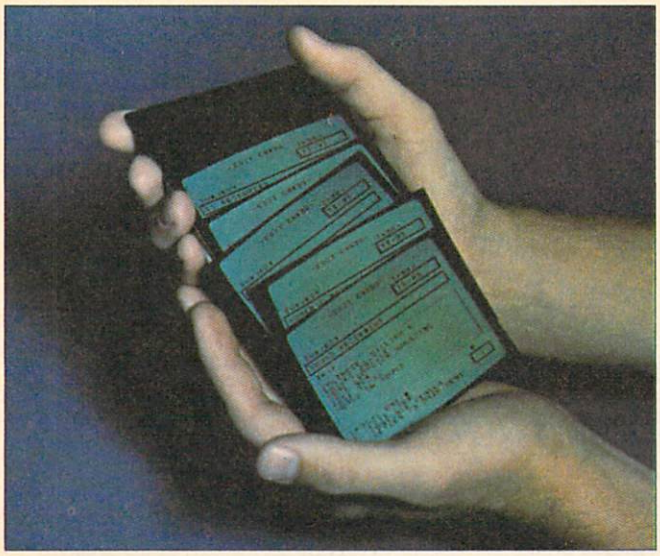
A couple of days ago, I was skimming through the magazine rack and saw issue 5.5 with the small banner announcing Atari coverage. I immediately bought the issue. A day later I mailed my subscription.

I have owned an Atari for about two years now. I started with the 400 and the 410 recorder, but now have an 800XL, disk drive, modem, and printer.

Again, thanks for the expanded coverage. You can expect me to be a continued subscriber.

Dennis E. Rees
Willard, OH 44890

Glad we are able to fill your need, Dennis. We look forward to bringing Atari users a fresh—and badly needed—approach to home-computer usage. **HCM**



TI Card Shuffler

by Randy Thompson
HCM Staff

Shuffle your file cards? Actually, computerized card shuffling can help you organize and expand your filing capacity.

[Note: TI Card Shuffler is the second part of TI Card Trix, which appeared in HCM Vol. 5 No. 5.]

In our last issue, we introduced you to *TI Card Trix*—a program designed to help you create and edit an inventory of file cards containing thoughts, book references, recipes, etc. Now we'd like you to meet *Card Trix*'s helpful companion—*TI Card Shuffler*. This companion completes important filing functions faster than even the best secretary. And it lets you shuffle your cards between many different cross-referenced folders. The program begins by offering you two options:

SEARCH CARDS
EXIT PROGRAM

Search Cards

This option brings you to the heart of the program. You can begin indexing your cards from 4 different *Card-Trix* folders. When you select Search Cards for the first time, the program asks you to enter the search parameters you want to use to pull cards from folders. First, enter the item for which you are searching—any sequence of up to 28 characters. Then, specify the field(s) by which you want to search. You can search using the Index, Subject, and Text fields or any combination of the three fields. You must select at least one field, or the computer becomes irritated and beeps at you. Search Cards consists of these 4 options:

GET A FOLDER
TOSS A FOLDER
SET SEARCH PARAMETERS
RETURN TO MAIN MENU

Get A Folder—When you select Get A Folder, the program asks for the name of the folder that you want to search through. After you enter the appropriate file name, the computer searches through the folder, noting all cards that meet the search parameter specifications. The computer then displays the number of cards, if any, that it finds.

You can perform this indexing search with a maximum of 4 folders in memory. If you want to search through a fifth folder, you must create space by discarding one of the folders with the Toss A Folder option.

Toss A Folder—When you select this option, the computer displays a list of all the folders that are currently in memory. To toss one of the displayed folders, enter the number that appears next to the unwanted folder. Don't worry: Tossing a folder does not erase it from the disk or cassette—only from memory.

Set Search Parameters—You can change the search parameters with this option. If you can't find what you're looking for, for example, try changing the fields by which you are searching.

Return To The Main Menu—When you return to the main menu with folders in memory, the menu expands to offer you two more options:

PRINT FOLDER INDEX
SAVE INDEXED FOLDER

Print Folder Index

The folder index is a cross-reference listing of all the folders in memory. The printout contains the name of each folder in memory, the item used in the search for these folders, and numbers corresponding to the card(s) where the item was found. Let's say you printed a search for the word "DOG." Now, any time you need to find a card containing "DOG," simply refer to the indexed printout. You can also display a cross reference on-screen by selecting the Screen option. You can press the space bar to pause the screen display as it scrolls by.

Save Indexed Folder

Want a whole new folder instead of just an indexed printout? With this Save option you can *shuffle* cards from the 4 folders in memory into a new folder.

When making a new folder, the program asks you for the disks or cassettes that hold the cards you want to shuffle. Now, insert the destination disk or cassette, and computer will save the indexed folder to the device with the file name you choose. (*Note: Each folder holds no more than 25 cards; therefore, the program places only the first 25 cards it finds in the new folder.*)

Exit Program

If you select this self-explanatory option and forget to make a printout or perform a search, don't worry. The program automatically asks you if you're sure you want to exit. At this point, you can either re-enter or exit *Card Shuffler*.

HCM Glossary terms: cross-reference, index, parameter.

For your type-in listings, see HCM PROGRAM LISTINGS CONTENTS.

HCM



Electronic Postman

by Randy Thompson
HCM Staff

Neither rain nor snow nor the strictest of deadlines can keep you from writing individualized correspondence with this mail-merging program.

If you're like many people today, you probably write personalized correspondence only when you have to. It's a time-consuming task! And it doesn't help when your list of addresses and phone numbers grows exponentially year after year, listing friends, relatives, business associates, congressmen, fellow club members, and more.

Frustration usually sets in when you need to contact all or most of these people *simultaneously* for holiday greetings, party or wedding invitations, thank-you letters, meeting announcements, business memos, or political messages. Giving each letter that "personal touch" is almost impossible—especially around the holiday season when you've got so many other things to do.

If you could only hire someone to keep track of all the important information that you cram into your little black book—someone to help you write personalized form letters. Such a person would have to sort, search, and update your records very quickly and at a moment's notice. And this person would also have to type each letter, inserting the appropriate "personalizing" details.

Sounds like you'll have to hire a professional secretary—right? Not really. Your computer can compete with even the best secretary—if you have the *Electronic Postman*. When linked with *Electronic Typewriter* (published in Vol. 5, No. 5), *Electronic Postman* becomes a powerful "mail merger" data base, capable of quickly organizing, up-dating, and manipulating your entire address book.

Individual Form

Let's say you want to send all of your business associates a letter explaining a new policy that your company recently implemented. Use *Electronic Typewriter* to compose the form letter, and let *Electronic Postman* quickly insert specific names, addresses, and other individualizing information. There's no repetitive typing involved! The *Postman* prints each letter according to the format you specify.

The program's main menu offers the following 6 options:

EDIT FILE
SORT FILE
SEARCH FILE
LOAD/SAVE FILE
PRINT FILE
ELECTRIC MERGE

Edit File

This option allows you to create lists of up to 50 names and addresses or alter previously prepared lists. Each record consists of 7 different fields of information—Name, Address, City, State ZIP, Phone Number, Text, and Code (see Photo 1). A simple keypress moves your cursor to any of the 7 fields that you want to edit (see the Control Capsule for your machine). The first 5 fields are self-explanatory. You can use the Text field for additional information that you wish to include in the file, such as occupation, type of computer owned, birth date, spouse's name, or whether you'll be sending that person a Christmas card this year. See the Electric Merge section below for an explanation of the Code field.

A small box in the lower-right corner of the editing screen displays a number corresponding to the record that you're currently working on. From this box, you can jump to any record within the file, by entering the appropriate number and pressing [RETURN].

Each time you press [RETURN] in the Edit File option, the cursor advances one field. If you are editing the very last field—Code—a [RETURN] advances the program to the next record.

Sort File

You can use this option to arrange all your records alphabetically or numerically (according to the characters' ASCII representations). Simply choose one of the 7 fields to use as criteria for sorting your records. A sort by State, for example, puts your records in

Bug-Out Identifies Bugs
Dear Sir:

I own an Atari 800XL with a single disk drive, a tape recorder, and a very dotty matrix printer. I have previously owned four other computers, all of which were ignored, for the most part, by the publishing industry. It was really nice to see the support for my machine at last.

This reaction is tempered a little after a couple weeks of (intermittently) typing programs from your magazine! The first thing I typed in was the Proofreader, and I must admit that it worked. The second thing I undertook was the NanoProcessor and my experience was not so great.

The first problem arose when I tried to enter line 920, a DATA item with a clear and unmistakable Pounds-Sterling symbol. Atari X models have such a symbol; however, the use of it normally excludes the other graphic characters used in the NanoProcessor display. I eventually figured out that what you must have meant was a SHIFT + character used (by Atari mostly in other languages than BASIC) for mod division.

The second problem was rather more of a bug, and took a while to figure out. The program checks the location 764 for last key pressed, and branches accordingly. At least, it ought to; it doesn't. The problem seems to be well illustrated by the segment from line 1500 to 1580. The variable B contains the last-key-pressed identity from location 764; line 1520 tries to match it against the ASCII value of the control-panel keys. But the Ataris use three interlocking character set designations: ASCII; Internal code, which is the sequential position of the character in the shape table; and Keyboard code, which bears little resemblance to the other two. Keyboard code is tied to the row-column position of the keys, and it is a member of this set which is stored at 764. Here's a one-liner for you to illustrate the point:

```
1 FOR A=1 TO 256:A = PEEK(764):
?CHR$(A);'' '';:NEXT A
```

PLEASE type it EXACTLY as shown. The continuous reassignment of the counter variable value within the loop keeps the thing going forever, or until you have had enough (use BREAK to get out). Press any key, alone or with either or both of CONTROL and SHIFT keys. Try the Atari key (inverse toggle) and the CAPS, ESC, and RETURN keys. Try using each key with and then without the CONTROL. Try any key presses you like. CTRL-1 which pauses all action and then unpauses it, and the BREAK key will not give unusual results.

But back to the program bug. I thought of looking for the keys that would give the result, in effect reassigning the control panel in the game. Almost immediately after that, I thought of altering line 1580 so that the correct connection with ASCII would result. Both of these approaches had to be rejected because not all ASCII values represent valid keycodes. Not even all 6- or 7-bit numbers are valid. As a matter of fact, it turns out that some of the values required for the program are not valid K-codes. The only workable solution I found was to rewrite the following two lines.

```
1520 READ ZQ: IF B=ZQ THEN 1550
1580 DATA 49,50,51,52,60,62,80,69,6,
     73,82,76,62
```

I have not yet had a chance to play with the program, to see whether or not it is completely debugged. At least these fixes get it started running.

I look forward to the next few issues of your magazine. I hope it lives up to the high promises made by issue 5.5, apart from these details.

Diane Hoffman
Oshkosh, WI 54901

We're glad to have such a knowledgeable Atari reader, Diane. Thanks for the comments. The first bug you noted was due to an oversight in our process for translating the program to our type-set listing format. The other bug seems to stem from an error you apparently made in typing in the program. Line 1460 does indeed PEEK location 764, but B does not contain the value from that location—it contains the ATASCII returned from the GET #2, B statement. Thus, that aspect of the program works correctly as published. Thanks, though, for the excellent discussion of the various ways Atari machines represent key input, as well as for the neat little "one-liner."

Word Pro From ProDOS To 3.3
Dear Sir:

I am having some problems with my Apple *II*c. A few weeks ago, a friend of mine sent some documents he created with his word processor to me. The documents were on a ProDOS disk and were stored as text files. My problem is that most of the documents he sent to me that were stored as text files were stored under a sub-directory. My word processor can read text files when they are directly stored as text files, but not when they are stored under a sub-directory. My word processor translates the text files to files my word processor can understand. (The files are on a DOS 3.3 disk now because that is the only format my word processor can translate them from. I copied the files from the ProDOS disk. This procedure has always worked until now.) Do you know of any program or procedure that would allow me to store these text files directly onto the disk instead of under a sub-directory so I could print the files on my printer with my word processor? I need a solution really soon.

Also, I have heard about a Z-RAM board by Applied Engineering that would allow you to add 256 or 512K of additional memory and a Z-80 microprocessor to the Apple *II*c. Their ad says that it comes with a RAM disk so programs could run 20 times faster. It also said that you could run regular Apple software and CP/M software without unplugging the board. Do you think you could review this product? It sounds very interesting.

Thank you very much for your help.

Pete Sison
San Diego, CA 92139

In order to convert a file from ProDOS to DOS 3.3 you need two disk drives and the ProDOS Convert utility from Apple. If you have these, the various sub-directories shouldn't give you any trouble in conversion—just specify the complete ProDOS pathname in the convert program. We are evaluating the Z-RAM card from Applied Engineering and will be reviewing it in a future issue.

Hard-Driving Junior?
Dear Sir:

Like thousands of others, I thoroughly enjoy your magazine. In particular, I have used your techniques to successfully add a second floppy drive to my PCjr. Now I wonder about adding a hard drive. With the rapid de-escalation in hard drive prices this would appear to be a valuable next step. Any possibilities of your experts telling us how to do that? I am sure a number of us would benefit greatly.

Warren T. Dent
Indianapolis, IN 46260

Warren, our experts are currently working on hard-disk drive solutions for the PCjr, so stay tuned to HCM for cost-effective means to expand your PCjr's capabilities. Also, if any other readers desire additional Junior enhancements, please let us know so we can work toward giving you what you need.

Terminal Tape Troubles
Dear Sir:

I recently purchased a modem through mail order for my Commodore 64. Somehow there was a mixup in my order and the company I ordered from sent me the terminal software on disk instead of cassette tape as I requested. I contacted the company and they told me that the terminal software was unavailable on tape. When I borrowed a friend's disk drive to load the program and try to transfer it to tape, I found that I could only access one line of the program which was a SYS command. Is there any way I can access the full terminal program and copy it to tape or will I be forced to purchase a disk drive to use my modem?

Matthew Correll
Grinnell, IA 50112

Buying a disk drive is not a bad idea Matthew—especially now that the price on the Commodore 1541 is so low—but it is not the only answer. Several fine terminal programs come on tape and even cartridge. The popular VIP Terminal from Softlaw is available on tape. The SkiWriter II cartridge from Prentice-Hall (Reviewed in HCM Vol. 5 No. 4) is both a word processor and terminal program. As for copying and modifying the program you have now, don't try. It may be copy protected, and the attempt may void its warranty. If what you have does not work with your current system, we suggest that you either buy a disk drive or return the package.

Mini-Amnesia On Mini-Memory
Dear Sir:

The Mini-Memory module for my TI home computer will not accept assembly programs from tape, or BASIC programs from the computer. I assume the battery has died.

Could you tell me if this can be remedied and if so, where do I send it? How much would it cost? How long would it take? Have I diagnosed the problem correctly?

Any information you could send me would be appreciated.

John Wood
Salem, OR 97306

John, it's good that you should bring up this subject. There are a lot of Mini-Memory car

tridges out there with weak or failing batteries. There's a very good possibility that your Mini-Memory battery has finally gone to that great recharger in the sky. The symptoms you describe fit the dead-battery syndrome, though that doesn't rule out the possibility that the connector could be dirty or the cartridge misaligned so that the pins on the card edge don't fit correctly when the cartridge is inserted into the system. Fortunately, there is help on the way. We are currently tracking down the least-expensive sources for the battery, and will tell you how to replace it in next issue's "Home Computer Tech Note" for TI.

Personal Accounts-Payable?
Dear Sir:

I have an IBM PC and I am looking for a computer program that will do the following things. I was wondering if you would have this on any of the disks that are available through you, or if you might know where I could find it.

It is to manage my personal bills. I would like to be able to enter the company name, company address, or a code number for each company, subtract the payment made and have the balance show. The next month I would like to be able to subject the payment again and have the account balance show. It would work like an accounts payable file.

Thank you for your help.

Florence M. Glenn
Santa Ana, CA 92702

Florence, so far we have not published a program that will totally "fill the bill" for you. We are, however, developing such a program—one that will not only allow the kind of file you suggest, but also provide great flexibility in customizing other kinds of alphanumeric/calculating data files. Look for this software soon in an upcoming issue.

Looking For Artificial Intelligence
Dear Sir:

I will be a sophomore in high school next year and have acquired (through certain channels) a free hour each day which I spend working with Apple IIe computers. I also have a Commodore 64 at home and I am planning to buy the new C-128 when it arrives on the market.

My question is this: Do you know of any companies that specialize in literature and/or software on artificial intelligence programming? I intend to focus my future computer studies on this particular subject.

Thank you.

Philip G. Jones
Slater, MO 65349

Philip, your local library will probably have a few major books on Artificial Intelligence (AI), although such books are largely non-technical and are targeted for the popular market. Most famous among these is The Fifth Generation, *by Feigenbaum and McCorduck, which talks about Japan's push for AI. Others include:* Artificial Intelligence, *by L. Stevens,* The Thinking Computer, *by B. Raphael, and* Godel, Escher, Bach, *by D. Hofstader. If you want technical information and news of the current state-of-the-art, a newsletter published out of Scottsdale, Arizona, called "AI Trends," will fill your needs. You will find that most AI research these days, centers on "expert systems"—software that provides the kind of technical advice usually reserved for experts in a given field. For example, Hewlett-Packard employs an expert system on the assembly line to help diagnose defects in the structure of microchips. These programs use a combination of raw data bases and a "knowledge" base garnered by quizzing engineers on every conceivable "what-if" situation. Some of these professional techniques have trickled down to the microcomputer level—with programs like Personal Consultant for the TI-99/4A, Experlisp for the Macintosh, and the Knowledge Engineering System for the IBM PC. Human Edge Software offers a few such "decision making" programs for the Apple II family; but we don't know of any available for the C-64 or the C-128. (The C-128 should be compatible with AI software written for CP/M systems, however.) As these kinds of programs become more applicable to home use, we will certainly cover them in HCM.*

A Pound Sign For A Backslash
Dear Sir:

Imagine my surprise. I have never looked in your magazine, just at the cover. This time—Atari!

Your magazine seems much more imaginative than most of the others, especially in your programs and explanation sections. I see an excellent relationship growing.

However, is it just because you are new to the Atari field? Your NanoProcessor program in Vol. 5, No. 5 has a bug that can't be just a typo. Line 920 has a data statement which includes an English pound sign! I can access this symbol on my 800XL with the International Character function, but it reverts to a simple CONTROL 'H' graphics symbol on LISTing or RUNning. What did you have in mind?

Second, I can't get the program to work. I have CAREfully checked my listing but when I RUN the program I get an Input Statement Error (#8) on line 820. For some reason my computer will not accept the line 920 data statements. If I remove the offending statements (losing the switch pointer) the program runs for awhile until I enter the third line of the sample program where I get a String Length Error (#5) in line 1660.

Last, but not least, Atari Bug-Out doesn't DO anything! It runs, it asks for output device, it says READING, but it doesn't access my cassette file. Is it supposed to operate my cassette recorder? Your instructions don't say!

I know this letter looks hyper-critical, but it's not meant that way. I'm super impressed with your first Atari issue. I know it can only get better.

Norm Palosky
Alexandria, VA 22304

Thanks for the input, Norm, and welcome aboard. For the answer to the first problem you mentioned, see "Bug-Out Identifies Bugs" (above). As for the Bug-Out program, other readers have experienced no difficulty; we can only conjecture that you have made a typing error.

No ProDOS-Compatibles
Dear Sir:

I have three different Apple II+ compatible computers (from two different manufacturers) that I have been using heavily for over two years. I haven't had the slightest compatibility problem with loads of different software and with different operating systems, including CP/M and Pascal. But ProDOS doesn't work. None of the ProDOS diskettes that you have sent me works. Neither does one recently purchased program which is on a ProDOS diskette. The disks boot up displaying the title screen for the ProDOS release, then the computer locks up. The only key that the computer will respond to after that is RESET, which causes more problems. (Note: Two of the computers have 16K RAMcards in slot 0, and the other has a 128K Saturn board.) Is it necessary to purchase a ProDOS preboot diskette? Or is there some multiple-letter code to type in? Or has Apple figured out a way to defeat the "compatibles" with this latest operating system?

If I'm out of luck with ProDOS, would it be possible for you to send me your Home Computer ON DISK software with DOS 3.3?

Mike Rosa
Peru, IN 46970

Mike, on the second page of the Table of Contents in this issue, there are instructions for running ProDOS programs on Franklin machines. You don't say which brand of Apple-compatible you have, so we can't guarantee this will work for you. In Vol. 5, No. 5 , a letter from Henry Curry provides another way of fixing a ProDOS disk for Franklin—by using a "Sector Editor."

Expansion Of Coverage Hailed
Dear Sir:

This is the first time I have ever written to any magazine and I just wanted to say thanks. I am an avid Atari user and will help support anyone who supports Ataris. The first time I saw your magazine was several months ago on the newsstand. I was disappointed to see that you did not include Atari in your coverage.

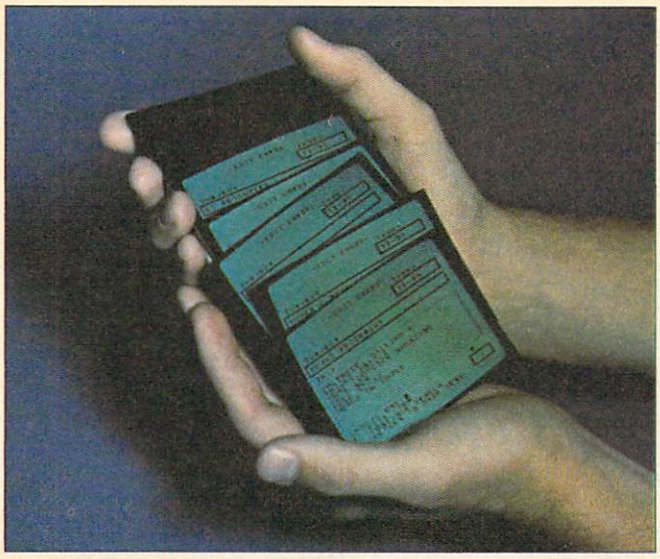
A couple of days ago, I was skimming through the magazine rack and saw issue 5.5 with the small banner announcing Atari coverage. I immediately bought the issue. A day later I mailed my subscription.

I have owned an Atari for about two years now. I started with the 400 and the 410 recorder, but now have an 800XL, disk drive, modem, and printer.

Again, thanks for the expanded coverage. You can expect me to be a continued subscriber.

Dennis E. Rees
Willard, OH 44890

Glad we are able to fill your need, Dennis. We look forward to bringing Atari users a fresh—and badly needed—approach to home-computer usage. **HCM**



TI Card Shuffler

by Randy Thompson
HCM Staff

Shuffle your file cards? Actually, computerized card shuffling can help you organize and expand your filing capacity.

[Note: TI Card Shuffler is the second part of TI Card Trix, which appeared in HCM Vol. 5 No. 5.]

In our last issue, we introduced you to *TI Card Trix*—a program designed to help you create and edit an inventory of file cards containing thoughts, book references, recipes, etc. Now we'd like you to meet *Card Trix*'s helpful companion—*TI Card Shuffler*. This companion completes important filing functions faster than even the best secretary. And it lets you shuffle your cards between many different cross-referenced folders. The program begins by offering you two options:

SEARCH CARDS
EXIT PROGRAM

Search Cards

This option brings you to the heart of the program. You can begin indexing your cards from 4 different *Card-Trix* folders. When you select Search Cards for the first time, the program asks you to enter the search parameters you want to use to pull cards from folders. First, enter the item for which you are searching—any sequence of up to 28 characters. Then, specify the field(s) by which you want to search. You can search using the Index, Subject, and Text fields or any combination of the three fields. You must select at least one field, or the computer becomes irritated and beeps at you. Search Cards consists of these 4 options:

GET A FOLDER
TOSS A FOLDER
SET SEARCH PARAMETERS
RETURN TO MAIN MENU

Get A Folder—When you select Get A Folder, the program asks for the name of the folder that you want to search through. After you enter the appropriate file name, the computer searches through the folder, noting all cards that meet the search parameter specifications. The computer then displays the number of cards, if any, that it finds.

You can perform this indexing search with a maximum of 4 folders in memory. If you want to search through a fifth folder, you must create space by discarding one of the folders with the Toss A Folder option.

Toss A Folder—When you select this option, the computer displays a list of all the folders that are currently in memory. To toss one of the displayed folders, enter the number that appears next to the unwanted folder. Don't worry: Tossing a folder does not erase it from the disk or cassette—only from memory.

Set Search Parameters—You can change the search parameters with this option. If you can't find what you're looking for, for example, try changing the fields by which you are searching.

Return To The Main Menu—When you return to the main menu with folders in memory, the menu expands to offer you two more options:

PRINT FOLDER INDEX
SAVE INDEXED FOLDER

Print Folder Index

The folder index is a cross-reference listing of all the folders in memory. The printout contains the name of each folder in memory, the item used in the search for these folders, and numbers corresponding to the card(s) where the item was found. Let's say you printed a search for the word "DOG." Now, any time you need to find a card containing "DOG," simply refer to the indexed printout. You can also display a cross reference on-screen by selecting the Screen option. You can press the space bar to pause the screen display as it scrolls by.

Save Indexed Folder

Want a whole new folder instead of just an indexed printout? With this Save option you can *shuffle* cards from the 4 folders in memory into a new folder.

When making a new folder, the program asks you for the disks or cassettes that hold the cards you want to shuffle. Now, insert the destination disk or cassette, and computer will save the indexed folder to the device with the file name you choose. (*Note: Each folder holds no more than 25 cards; therefore, the program places only the first 25 cards it finds in the new folder.*)

Exit Program

If you select this self-explanatory option and forget to make a printout or perform a search, don't worry. The program automatically asks you if you're sure you want to exit. At this point, you can either re-enter or exit *Card Shuffler*.

HCM Glossary terms: cross-reference, index, parameter.

For your type-in listings, see HCM PROGRAM LISTINGS CONTENTS.

HCM



Electronic Postman

by Randy Thompson
HCM Staff

Neither rain nor snow nor the strictest of deadlines can keep you from writing individualized correspondence with this mail-merging program.

If you're like many people today, you probably write personalized correspondence only when you have to. It's a time-consuming task! And it doesn't help when your list of addresses and phone numbers grows exponentially year after year, listing friends, relatives, business associates, congressmen, fellow club members, and more.

Frustration usually sets in when you need to contact all or most of these people *simultaneously* for holiday greetings, party or wedding invitations, thank-you letters, meeting announcements, business memos, or political messages. Giving each letter that "personal touch" is almost impossible—especially around the holiday season when you've got so many other things to do.

If you could only hire someone to keep track of all the important information that you cram into your little black book—someone to help you write personalized form letters. Such a person would have to sort, search, and update your records very quickly and at a moment's notice. And this person would also have to type each letter, inserting the appropriate "personalizing" details.

Sounds like you'll have to hire a professional secretary—right? Not really. Your computer can compete with even the best secretary—if you have the *Electronic Postman*. When linked with *Electronic Typewriter* (published in Vol. 5, No. 5), *Electronic Postman* becomes a powerful "mail merger" data base, capable of quickly organizing, up-dating, and manipulating your entire address book.

Individual Form

Let's say you want to send all of your business associates a letter explaining a new policy that your company recently implemented. Use *Electronic Typewriter* to compose the form letter, and let *Electronic Postman* quickly insert specific names, addresses, and other individualizing information. There's no repetitive typing involved! The *Postman* prints each letter according to the format you specify.

The program's main menu offers the following 6 options:

EDIT FILE
SORT FILE
SEARCH FILE
LOAD/SAVE FILE
PRINT FILE
ELECTRIC MERGE

Edit File

This option allows you to create lists of up to 50 names and addresses or alter previously prepared lists. Each record consists of 7 different fields of information—Name, Address, City, State ZIP, Phone Number, Text, and Code (see Photo 1). A simple keypress moves your cursor to any of the 7 fields that you want to edit (see the Control Capsule for your machine). The first 5 fields are self-explanatory. You can use the Text field for additional information that you wish to include in the file, such as occupation, type of computer owned, birth date, spouse's name, or whether you'll be sending that person a Christmas card this year. See the Electric Merge section below for an explanation of the Code field.

A small box in the lower-right corner of the editing screen displays a number corresponding to the record that you're currently working on. From this box, you can jump to any record within the file, by entering the appropriate number and pressing [RETURN].

Each time you press [RETURN] in the Edit File option, the cursor advances one field. If you are editing the very last field—Code—a [RETURN] advances the program to the next record.

Sort File

You can use this option to arrange all your records alphabetically or numerically (according to the characters' ASCII representations). Simply choose one of the 7 fields to use as criteria for sorting your records. A sort by State, for example, puts your records in

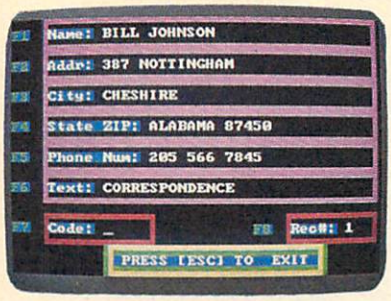


Photo 1
(From the IBM version) Each record contains 7 fields of information that you can edit to keep your information up-to-date.

alphabetical order according to the letters in each state's name. A sort by Phone Number places them in numerical order. To keep your records in order at all times, conduct a sort every time you add any records to your file.

Search File

This option helps you find any record in your file by searching through each record for a sequence of characters that you designate. Sound confusing? It's actually quite simple. When you enter the Search File option, the program asks you to enter the field you want to use as a search base. It then asks you to enter the sequence of characters to use in the search. For example, to find everyone in your file whose ZIP code is 97401, simply select State ZIP and type in "97401." Or if you remember only part of your barber's last name—you know it begins with "Mac"—don't page through all the records until you stumble across the right one. Perform a name search for "Mac," and let the computer do the work.

Electronic Postman displays any record it finds, then offers the following three choices: You can edit the record, continue the search, or exit to the main menu. If *Electronic Postman* cannot find an item, it tells you so and brings you back to the main menu.

Figure 1
Codes For Electric Merge

Code	Function
*N *	Full name
*F *	First name
*L *	Last name
*A *	Address
*C *	City
*S *	State ZIP code
*P *	Phone number
*T *	Text

Load/Save File

This option does exactly what its name implies. It lets you save and load files (each containing up to 50 records).

Print File

To get a printout of your file, select this menu option. The program asks you to input the numbers corresponding to the first and last records you want printed. For example, if you enter "1" as the first record number and "34" as the last, the program prints records 1 through 34. For alphabetized printouts, sort your files prior to printing.

Electric Merge

If you missed last issue's *Electronic Typewriter* program, you should seriously consider ordering it as a companion to the *Electronic Postman*. The Electric Merge option in the *Postman* integrates these two "electronic" programs, creating a very powerful writing tool.

By inserting special codes in your *Electronic Typewriter* text, you can define locations to place specific pieces of *Electronic Postman* data, such as names, addresses, and phone numbers. (See Figure 1 for a list of codes.) For example, if you need to send identical letters to several different people, you can

Figure 2
Sample *Electronic Typewriter* File

```
Dear *F        *,
Thank you so very much for the wonderful
wedding gift. Of all the presents we
received, the *T          * is one of the
most thoughtful and useful.
We are glad you could make it to the
reception. The food and spirits couldn't
have been better--especially the spirits.
We hope the weather in *C      * is nice
this time of year. Honolulu was a little
too warm for us, so we spent most of our
honeymoon on the relaxing and romantic
island of Maui.
Be sure to call or write, and let us know
if you move from *A                * or
change your phone number from *P      *.
Next time we are in *C      *, we'll be
sure to look you up. Thank you again for
the *T          *.
Sincerely,

Sigmund and Gertrude Green
```

Figure 3
Same Sample File After Electric Merge

```
Dear Herbert,
Thank you so very much for the wonderful
wedding gift. Of all the presents we
received, the toaster oven is one of the
most thoughtful and useful.
We are glad you could make it to the
reception. The food and spirits couldn't
have been better--especially the spirits.
We hope the weather in Antelope is nice
this time of year. Honolulu was a little
too warm for us, so we spent most of our
honeymoon on the relaxing and romantic
island of Maui.
Be sure to call or write, and let us know
if you move from 3872 Roosevelt St. or
change your phone number from 123-4567.
Next time we are in Antelope, we'll be
sure to look you up. Thank you again for
the toaster oven.
Sincerely,

Sigmund and Gertrude Green
```

personalize each letter by inserting a name code into your text. When the Electric Merge encounters this code, the program replaces the code with the name you have specified from the *Electronic Postman* file. Each time you merge an *Electronic Typewriter* text file with an *Electronic Postman* record, the program outputs the result to the printer. (NOTE: All codes must be uppercase.)

In order to make *Electronic Typewriter* as speedy as possible, we have included no word-wrap feature. Therefore, you must specify the number of characters you want to merge from the *Postman* record to the *Typewriter* text. For example, if the distance between the first and last asterisk in the code is 5 spaces, then that code can hold a maximum of 5 characters. When a piece of data, such as a first name, takes up less than the defined code length, the program adjusts the entire line of text, leaving no unnecessary spaces between words. However, if the data is longer than the maximum defined length, the program truncates the data to fit. Therefore, if you allot only 5 coding spaces for a 7-character name, the program lops off the last two characters in the name. Also note that the program won't separate first and last names, unless you enter them into your record in one of the following formats: 1) Redding, Herbert or 2) Herbert Redding.

For an example of how to use Electric Merge codes within your *Electronic Typewriter* text, see Figure 2. This example employs 5 different codes: First Name, Address, City, Phone Number, and Text. Figure 3 shows the printout of the merged text using data from an *Electronic Postman* record.

Starting To Merge

When you select the Electric Merge option from the main menu, the computer asks you for the name of the *Electronic Typewriter* text file you want merged. (This merge routine assumes that the *Electronic Postman* data is already in memory.) *Electronic Postman* has no option that enables you to load a template file, so if you want formatted printouts (centered, right-aligned, or indented lines), save your text file with a template.

The program then asks if you want a Selective Merge, which allows you to choose specific records that you want merged with text from *Electronic Typewriter*. When you selectively merge, the program asks you to enter a password: the characters in the Code field of the record that you wish to merge. For example, if you enter "A1A1" as your password, the Electric Merge routine merges only the records with A1A1 in their Code fields.

You can use question marks as "wild cards" in the merge routine. For example, if you enter "FL?P" as a password, the program merges records with similar codes, such as FLIP, FLOP, or FL2P—but not SLIP. A password containing only question marks is the same as no password.

Once you've entered the appropriate password, the program begins merging your text and files. As each line of text is merged, the program sends it to the printer. It's a slow process, but it's much faster and far more accurate than manually typing the information. (Note that *Electronic Postman* does not merge records that have nothing entered into the Name field.) After the merging is complete, the program automatically returns to the main menu. Now, you're ready to begin another correspondence!

HCM Glossary terms: ASCII, field, file, merge, record.

For your type-in listings, see HCM PROGRAM LISTINGS CONTENTS.

HCM

CONTROL CAPSULE

Electronic Postman

Key	Function
ESC	Exit current mode
CONTROL N	Edit Name field
CONTROL A	Edit Address field
CONTROL C	Edit City field
CONTROL S	Edit State ZIP field
CONTROL P	Edit Phone Number field
CONTROL T	Edit Text field
CONTROL X	Edit Code field
CONTROL R	Change record number
CONTROL B	Erase current field
CONTROL I	Insert blank into current field
CONTROL D	Delete character under cursor
RETURN	Move to next field

CONTROL CAPSULE

Electronic Postman

Key	Function
ESC	Exit current mode
CONTROL N	Edit Name field
CONTROL A	Edit Address field
CONTROL C	Edit City field
CONTROL S	Edit State ZIP field
CONTROL P	Edit Phone Number field
CONTROL T	Edit Text field
CONTROL D	Edit Code field
CONTROL R	Change record number
RETURN	Move to next field

CONTROL CAPSULE

Electronic Postman

Key	Function
—	Exit current mode
F1	Edit Name field
F2	Edit Address field
F3	Edit City field
F4	Edit State ZIP field
F5	Edit Phone Number field
F6	Edit Text field
F7	Edit Code field
F8	Change record number
RETURN	Move to next field
SHIFT CLR	Erase current field

CONTROL CAPSULE

Electronic Postman

Key	Function
ESC	Exit current mode
F1	Edit Name field
F2	Edit Address field
F3	Edit City field
F4	Edit State ZIP field
F5	Edit Phone Number field
F6	Edit Text field
F7	Edit Code field
F8	Change record number
F10	Erase current field
ENTER	Move to next field

Cornucopia CATALOG™

Special Edition Winter '85/'86









Distinctive Products For Music & Electronics



Welcome To Cornucopia™

Welcome. I'm Gary Kaplan, publisher of *Home Computer Magazine, Music & Electronics Magazine,* and the *Cornucopia Catalog* you are now reading. If you haven't already done so, may I suggest that you play the enclosed (bound-in) Soundsheet on an available stereo turntable system by carefully removing the recording at the perforations, and placing it on top of a regular LP record for stability. You're in for a pleasant surprise . . .

Now that you've heard our musical extravaganza—and, I hope, enjoyed it—please take a tour of the catalog itself. We've included special informative "side bars" (shaded gold) on several of the pages to help you more fully understand some electronic music and audio concepts—information that is important for wise product selection.

If you find the catalog and recording interesting, I hope you will share them with your friends, relatives, and associates. And, if you know of someone who would be especially interested in our product offerings, drop us a line with their name and address. We'll be happy to mail them their own catalog.

Happy Shopping



Cornucopia™

Guarantees Your Complete Satisfaction 4 Big Ways

1. ORDER RISK-FREE WITH A 30-DAY RETURN PRIVILEGE

We want you to try anything in this catalog risk-free. If not satisfied—for whatever reason—simply return the product to us prepaid and insured, in new unmarked condition with all its original parts, instructions, packing materials, and our original invoice within 30 days of when you receive it. You will receive a courteous refund.

2. ORDER WITH COMPLETE PRICE-PROTECTION ASSURANCE

Cornucopia strives to offer the finest quality products at the best prices. You won't see the same item advertised for less elsewhere. In the unlikely event that you find any product you buy from us currently advertised, in stock, and selling for less than our catalog price, just send us a copy of the newspaper or magazine advertisement that appears during your 30-day risk-free trial, and a copy of your invoice. We'll issue you a courteous merchandise credit for the difference in price—or, at our option, we will refund the purchase price upon return of the merchandise, plus send you a free gift for your trouble.

3. ORDER WITH CONFIDENCE IN PRODUCT DURABILITY

We make sure every product is backed by a reputable manufacturer's service center. If you're having a problem geting satisfactory service—in or out of warranty—call our Customer Relations Representatives. We'll assist you in getting your item fixed or replaced in a reasonable time.

Your Satisfaction Is Our Business.℠

4. ORDER WITH COMPLETE PEACE OF MIND— KNOWING THAT EXPERT TECHNICAL SUPPORT IS AVAILABLE

Cornucopia is part of the Emerald Valley family of companies (including *Music & Electronics Magazine, Home Computer Magazine,* and World Class Audio). Before choosing a product for the Cornucopia Catalog, it is first carefully evaluated by product specialists at our Corporate Research Center, then put into daily use by our magazines' technical editors. All Cornucopia customers are welcome to direct their technical questions concerning the use of purchased products to our Technical Support Representatives. Many of the products that we sell are also supported by editorial coverage in *Music & Electronics Magazine*. Additional support is provided by our Corporate Data Processing Center through targeted mailings of the Cornucopia Product Awareness Letter™. This personalized newsletter keeps individual customers abreast of newly released products which have the potential to interface well with products they've already purchased. When you buy an item from the Cornucopia Catalog, we want to make absolutely sure you know how to use it properly, and continue to learn of new ways to fit it into evolving systems.

—GIFT CERTIFICATES—

Choosing the right gift for a musician or music lover is not an easy task. Why not make it easy by giving a Cornucopia Gift Certificate . . .

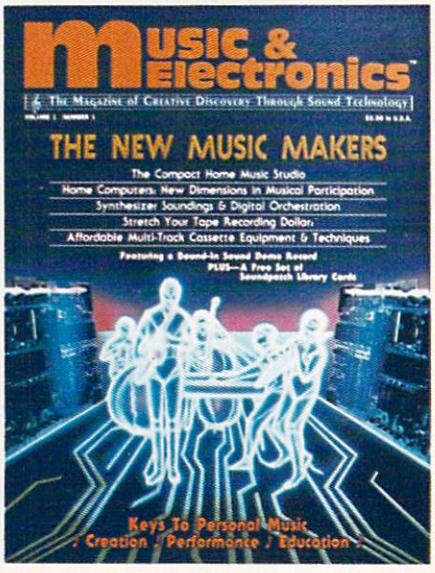
They're available in any denomination of your choice. Both your name and the recipients are hand-written on a fine card stock.

Surprise someone special, today!


THE CORNUCOPIA
Product Awareness Letter™
—Your Consumer P.A.L.



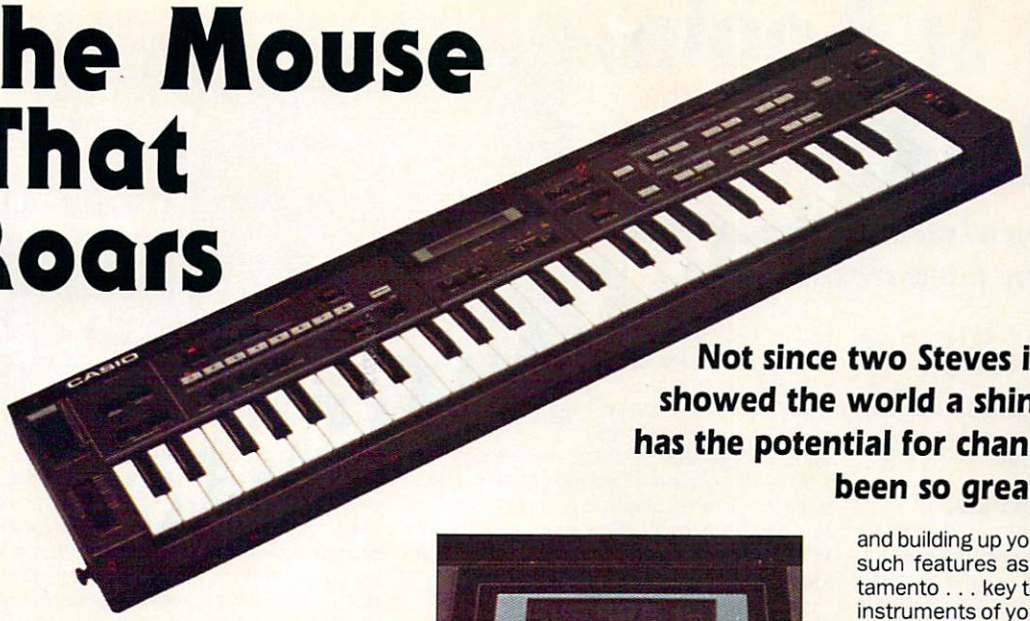
The Soundsheet takes you on an exciting journey into the world of electronically produced music.



Each issue of *Music & Electronics Magazine* explores many fascinating subject areas including introductory music theory, audio and digital recording techniques, and the interfacing of musical devices with home computers and home entertainment systems.

The entire contents of this catalog Copyright © 1985, Emerald Valley Publishing Co. Cornucopia, Cornucopia Catalog, Cornucopia Product Awareness Letter, Home Computer Magazine, Music & Electronics, World Class Audio, Your Consumer P.A.L., and Your Satisfaction Is Our Business are all trademarks or service marks of Emerald Valley Publishing Co. Dolby is a trademark of Dolby Laboratories Licensing Corporation; and dbx is a trademark of dbx Inc.

The Mouse That Roars



Not since two Steves in a garage showed the world a shiny new Apple has the potential for changing the world been so great!

This was the amazing little keyboard synthesizer that convinced us a major revolution was afoot. Casio, the consumer electronics giant, had quietly introduced another puny-looking musical keyboard—only this time, it turned out to be a professional quality electronic instrument of mammoth sound capabilities. Its tiny proportions, unpretentious parentage, and unbelievably low price—one step removed from the "toy keyboards" of the past—didn't fool the pros for long. Once a few "big names" in the music world got their hands on this little box, its powerful voice roared forth for all to hear . . . The secret couldn't be contained. In the blink of an eye, the state-of-the-art in musical electronics had taken a quantum leap. The high-cost barrier blocking the entry into professional music creation and performance had been torn asunder. The music world would never be the same again . . .



The informative LCD window displays all waveform and special effects parameters, and guides you in creating unique sounds of your own.

HIGH-TECH MADE PLAYER-FRIENDLY

The CZ-101 is the first electronic keyboard to provide the crystal clean sounds of digital synthesis for this incredibly low price. Its sophisticated Phase Distortion sound generation circuitry is organized into the simple block structure of an analog synthesizer with compare/recall capability—making it extremely easy to create and modify complex waveforms. The CZ-101's 8-stage sound envelope settings with 2 separate-but-mixable sound lines let you reproduce fat, multi-dimensional sounds with astonishing realism.

Internal memory stores 2 banks of 16 sounds each. The first bank is loaded with factory presets (in ROM) that cannot be overwritten. The second bank is for user storage and retains its memory (battery-backed RAM) even when the keyboard's power switch is shut off. An optional plug-in RAM cartridge expands the on-board storage capacity to a total of 48 different sound patches. These battery-backed memory cartridges provide convenient off-line storage for user-created sound libraries that load instantly.

You'll love the CZ-101's "high-tech" panel, with bright LED indicators above each function. A large LCD window display in the center of the front panel guides you through dozens of parameters for altering standard sounds and building up your own special effects (with such features as pitch bend, vibrato, portamento . . . key transposition) and creating instruments of your own design. Interacting with this display has been made virtually "foolproof."

4 FOR THE PRICE OF 1

The surprises don't stop with the CZ-101's spectacular sounds. This impressive synthesizer has versatile MIDI (Musical Instrument Digital Interface) capabilities. MIDI IN and OUT jacks on the back allow your CZ-101 to communicate with an exciting array of MIDI products that can greatly expand your instrument's capabilities. Such MIDI equipment—including other synthesizers for blending different sounds—can be added at any time, so your CZ-101 always remains a useful instrument in an expandable music creation and performance system.

If you own a MIDI sequencer, a simple press of the CZ-101's MIDI button, transforms this 8-voice polyphonic synthesizer into a 4-voice, multi-timbral expander. In this mode, the CZ-101 allows you to perform 4 completely different monophonic instrument sounds at once. Create a whole band—program voice 1 to be your bass, voice 2 to be your guitar, voice 3 to be strings, and voice 4 to be horns. Virtually any MIDI sequencer is compatible—see details in this catalog about the Casio SZ-1 MIDI Sequencer and MIDI software packages available for your home computer.

FOR ACTIVE PLAYERS ON THE GO

The CZ-101's small size, 7-pound weight, professional voice, and battery operation make this instrument "at home" in any environment. The headphone jack and line-out (for easy hook-up to your stereo or optional powered speaker) provide great flexibility whether at home or on the road. (Headphones allow the kids in the back seat to practice QUIETLY on those long road trips.) Comes complete with batteries and audio connection cord.

CZ-101 Suggested Retail Price $499.00

Cornucopia Price
CAT. NO. 1CCZ101 **$349** ($9.00 P&H)

Optional AC Adaptor (AD-5), RAM Cartridge (RA3) and Powered Speakers (AS-10 or AS-20) are found on page C-4. For Sennheiser MS 100 headphones, see page C-13.

Same Great Sounds With Standard-Sized Keys!



The CZ-1000 comes packed with the same great features as a CZ-101, plus it sports standard-size keys with one of the best "feels" in the industry. Enjoy the CZ-1000's rich sounds, easy programmability, and MIDI capabilities—all identical to those of the CZ-101. Unique dust-proof buttons, shoulder strap accomodations, 12-pound weight, and 3-way power supply make this an ideal value in a performance keyboard that's easy to pack along with you. It uses any of the optional accessories that are available for the CZ-101.

CZ-1000 Suggested Retail Price $699.00

Cornucopia PRICE
CAT. NO. 1CCZ1000 **$525** ($13.00 P&H)

"Play It Again, RAM!"

"They laughed when I sat down at the keyboard to play my latest creation . . .



. . . but when I pushed the PLAY button, the big professional sound of my personal electronic orchestra performed my composition with impeccable precision—turning their chuckles and smirks into stares of amazement and admiration!"

YOUR ONE-FINGER'S BEST PAL

Imagine a device capable of transforming a single synthesizer into an ensemble of different instruments. Imagine a recording device that operates just like a simple tape recorder, but doesn't use any tape. Imagine a device capable of recording your performance at *one* tempo and later playing it back at *another* tempo—without altering the pitch of a single note! Imagine a device that plays back your performance in perfect rhythm, even though you recorded it very slowly—over brief sessions spanning several days—one finger at a time! This is the Casio SZ-1 multitrack digital sequencer—a "smart" microprocessor storage device that transforms the CZ-101, CZ-1000, or other MIDI-compatable synthesizers into creative power tools and performance machines of incredible versatility.

Unlike other stand-alone sequencers in its price range, the SZ-1 isn't limited to two tracks with constant "bouncing" of material to always free up one track for recording. Its four separate tracks allow you to modify data on any track—at any stage of the composing process.

MANUAL & REAL-TIME RECORDING

Enjoy the option of recording in either the manual mode (a maximum of 3,600 notes) or the real-time mode (1,800 note maximum). In the manual mode, the LCD window graphically depicts your music for easy editing. Erasing or inserting single notes is a breeze. Overdubbing additional MIDI-assignable tracks is faster than ever, thanks to the SZ-1's "instant reset" (return to the beginning) and built-in metronome.

The SZ-1 is compatible with MIDI synthesizers with (or without) velocity-sensitive keyboards. Hook it up to a true performance keyboard like the Akai AX-80, and the SZ-1 will capture your music's expressive dynamics! Or connect it to a MIDI-compatible drum machine and listen to your drums follow along in perfect "sync" with your synthesized tracks. A built-in cassette interface allows you to save your work to inexpensive cassette tape. Or, use an optional memory cartridge for instant loads. The battery powered SZ-1 comes with five "D" size batteries and two 5-foot MIDI cables.

SZ-1 Suggested Retail Price $399.00

Cornucopia Price **$299** ($7.00 P&H)
CAT. NO. 1CSZ1

Optional AC adaptor AD-1 and Memory cartridge RA-5 are shown below.



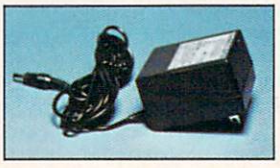
AS-10 Monitor—A powerful-yet-portable amplified speaker (5 watts) designed to use either batteries (6 D cells) or optional AD-5 AC adaptor.
Suggested Retail Price $79.95

Cornucopia Price **$69.95** ($5.00 P&H)
CAT. NO. 1CAS10



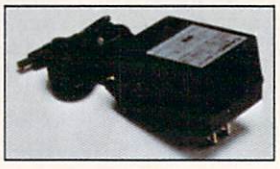
AS-20 Monitor—An amplified speaker (5 watts) with a built-in graphic equalizer and chorus with rate/depth controls. Works with either batteries (6 D cells) or optional AD-5 AC adaptor.
Suggested Retail Price $129.50

Cornucopia Price **$99.95** ($5.00 P&H)
CAT. NO. 1CAS20



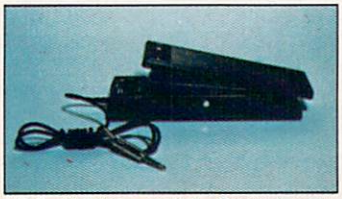
AD-1 AC Adaptor—Light-weight, economical plug-in power for the SZ-1 MIDI Sequencer.

Cornucopia Price **$12.00** ($3.00 P&H)
CAT. NO. 1CAD1



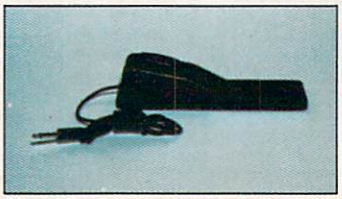
AD-5 AC Adaptor—Light-weight, plug-in power for the AS-10 and AS-20 speakers, and the CZ-101 or CZ-1000 synthesizer.

Cornucopia Price **$21.00** ($3.00 P&H)
CAT. NO. 1CAD5



VP-2 Volume pedal—Control your CZ-5000's volume using your foot as you play. Achieve crescendo and decrescendos effortlessly.

Cornucopia Price **$49.00** ($4.00 P&H)
CAT. NO. 1CVP2



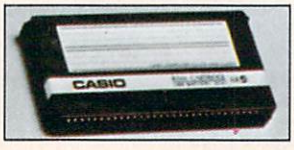
SP-1 Sustain pedal—Give your CZ-5000 that piano-like feel with this quality sustain pedal.

Cornucopia Price **$19.00** ($4.00 P&H)
CAT. NO. 1CSP1



RA-3 RAM Cartridge—Battery-backed memory cartridge (for all CZ keyboards) stores your sound patch libraries for quick-access loading.

Cornucopia Price **$39.00** ($3.00 P&H)
CAT. NO. 1CRA3



RA-5 RAM Cartridge—Battery-backed memory cartridge for SZ-1 sequence storage.
Suggested Retail Price $89.95

Cornucopia Price **$79.95** ($3.00 P&H)
CAT. NO. 1CRA5

Rambo's Real-Time Band



There's some hostile territory out there. A jungle of connection cords to come undone, booby-traps lying in wait for your special effects and splitter boxes, and an unexpected sync ambush for your keyboard arsenal . . .

When you pack up your gear for a crucial musical session, your mission's success demands peak performance from your equipment. That's why the CZ-5000 is fast becoming the choice of unstoppable pros determined to reach their musical goals no matter what the odds . . .

When Casio introduced the CZ-101, the product's price/performance ratio leapfrogged an entire generation of competitive products and established itself as a major technological milestone. (See our write up on page C-3.) Shortly thereafter, the SZ-1 MIDI sequencer was born (see page C-4). It was the perfect companion to the 101—matching it in price/performance and ease of use. After this one-two combination, Casio then delivered the knock-out punch with the CZ-5000, an integrated instrument that combined twice the power of its predecessors at the best price/performance ratio yet.

DOUBLE THE POWER—PLUS

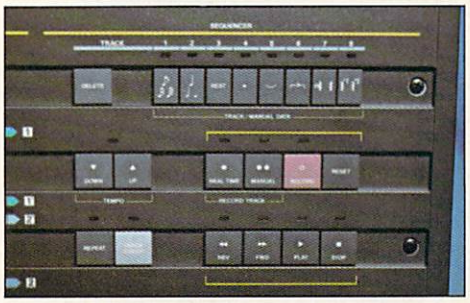
There's no arguing the fact that this is a stunning-looking machine. It draws oohs and aahs from all who see it . . .

At first glance, it appears that the CZ-5000 is simply two CZ-1000s and two SZ-1s—all within a full-size (5-octave) keyboard. But first glances are often deceptive. If that were all the difference between the units, it would still be more than enough to justify the difference in price: The CZ-5000 is simply a marvelous value.

When you use this keyboard in actual music production, however, you start to realize the subtle-yet-significant differences:

The CZ-5000 has the same standard-size keys as the CZ-1000 (the 101's big brother), but it boasts a 61-key, 5-octave—rather than a 49-key, 4-octave—range. Then there's the 16 big voices of the keyboard available in polyphonic mode—for fat, 8-part chords that sustain while hitting follow-ups (in real-time performance). Used as a MIDI expander, it can sound 8 different instruments—or any simultaneous combination of polyphonic sound(s) and monophonic sounds totaling 8 voices. For example, you can have a 3-note piano chord sounding while playing 5 other 1-note voices such as bass, brass, flute, organ, and hand-clap.

The special enhancements don't stop here. Your CZ-5000 also provides a keyboard-split capability with independent volume level settings—allowing you to, for example, play a synth bass with your left hand, and vibes with your right. Also, the standard CZ-series tone mix capability is a little more sophisticated—allowing independent level settings without having to go in and modify the actual patches. You can even program the built-in metronome to simulate different time signatures. Add in a stereo chorus capability, modulation wheel, stereo line-out jacks, and you have the makings of some very impressive special effects!



The built-in 8-track MIDI sequencer with a 6800-note capacity in manual mode and 3400-notes in real-time recording is as easy to use as a cassette tape recorder.

In addition to the built-in AC power supply, the unit has a built-in cassette tape interface for storing both sound patch data and sequencer data on standard cassette tapes, allowing you to build up economical libraries. The 5000 will also accept RA-3 RAM cartridges for storage of 16 sound patches in each.

All in all, this 26-pound handsome keyboard synthesizer has an impressive arsenal of features at an equally impressive price. When you attack any mission with this explosive performance package, you are fully armed and equipped to settle any score . . .

CZ-5000 Suggested Retail Price $1199.00

Cornucopia Price
CAT. NO. 1CCZ5000 **$849 ($21.00 P&H)**

Optional RAM Carttridge (RA-3), Casio powered speakers, VP-2 volume pedal, SP-1 sustain pedal, are on shown on page C-4. See page C-12 for Fostex powered monitors. For Sennheiser MS 100 stereo headphones, see page C-13.

WE'RE TALKIN' SYNTHESIZERS!

Synthesis is the combining of separate elements into a whole. A *synthesizer* is an electronic musical instrument used to create (*program*) tone colors, or timbres, from basic building blocks of sound (*waveforms*). Synthesizers typically come with factory-installed sounds (*presets*) stored in memory. You can either temporarily or permanently modify the presets, or concoct new sounds from scratch, using the resources of the instrument. The synthesizer's memory may be dumped onto a RAM cartridge or or standard cassette tape allowing you to save your sounds for future use. Over time, you can create a library of hundreds of these sounds which can be reloaded in groups of voices (*banks*).

Polyphonic synthesizers (capable of playing two or more notes together) are categorized by how many *notes* may sound at the same time. (A 6-*voice* synthesizer, for example, allows you to play up to 6 notes simultaneously.) A *multi-timbral* instrument is capable of producing more than one *tone color* at a time. For example, a multi-timbral synth could sound a flute melody over a string-bass rhythm pattern. This effect is limited only by the number of voices available on a particular instrument.

Certain synthesizers have a *stereo output*. On this type of instrument, the voices are distributed from left to right across the stereo field image in a variety of ways, depending on the design of the synthesizer. Some stereo instruments offer *chorusing*. This electronic effect enhances the sound by gently wavering the pitch and rolling the sound back and forth between a pair of speakers.

THE PIANO'S STEPCHILD

Typically, a synthesizer is equipped with a piano-like keyboard that permits you to play timbres as discrete pitches. Once electronic sounds are in pitch form, you may easily organize them into melody and harmony—the building blocks of music.

Some keyboards are *velocity sensitive*: the force with which you strike the keys affects the volume and tone of each note. Certain keyboards also offer *after touch*: when you continue to press a key after playing an initial note, the synthesizer measures this pressure and uses it to trigger volume, modulation, or filter circuits. This feature gives you expressive fingertip control over sound, by allowing you to add vibrato to a note, or control its volume and tone after the initial keystroke has occurred.

ANALOG VS. DIGITAL

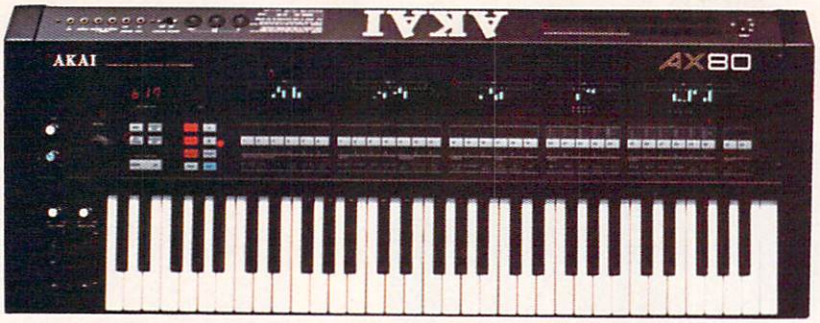
A synthesizer has the potential to produce a virtually limitless variety of timbres; it therefore has no particular sound of its own. Or does it?

A synthesizer is a sonic chameleon. Yet, like an acoustic instrument, each type of synthesizer has a characteristic sound quality. By and large, the method used to generate the raw waveforms employed in synthesis determines this overall sound. *Analog* synthesizers use continually varying voltages to control oscillators in a one-to-one (analog) correspondence to the audio signal. These instruments then filter and modulate the results to create new sounds. Recent developments in computer technology have now brought us *digital* synthesizers, which store sound as numbers. Analog synths are known for their warm, lush, buzzy timbres. Digital synths are recognized for their clear, bright, defined sounds. Some synthesizers combine elements of both types by placing voltage-controlled oscillators under the command of a micro-processor.

WHEN IS A SYNTH NOT A SYNTH?

A *sound sampler* isn't limited to the overall sound quality and combined circuitry built into a particular model of synthesizer. Samplers record "real sounds" and store them in digital form. These sounds are accessed by a keyboard—allowing you to play harp or guitar or even barking dogs from the same instrument. Strictly speaking, these instruments aren't synthesizers—yet they are playing an increasingly important role in electronically produced music.

A G-Note Gets The AX



An 8-voice synthesizer with touch-sensitive keyboard for under $1000? "No Way!"—they said. It took a sharp Akai AX to slash the price barrier . . .

For real-time recording and performing, a musician requires a keyboard that captures all the music's subtle dynamics corresponding to the velocity with which one strikes the key. These expressive dynamics are then transmitted to other synthesizers and/or stored (in a sequencer's memory) together with other standard MIDI data.

VISUAL EDITING & MORE

But affordable touch-sensitivity is not all the Akai AX-80 has to offer: Analog synthesizers like this used to have many levers to control the oscillators and other components. Like many other manufacturers, Akai put on a digital-controller "front-end"—but, unlike others, didn't depend on the pervasive LCD to display read-outs of the sound parameters. Instead, the AX-80 sports five attractive Fluorescent Display (FD) units on the front panel. These enable visual checking and monitoring of the sounds being generated—effectively reducing the sound programming time.

The AX-80 is a 8-voice polyphonic synthesizer with two main oscillators and a sub-oscillator for a rich sound. In addition to the 32 preset sounds, this 34-pound unit contains 2 memory banks for user storage of up to 64 additional sounds. A standard cassette interface allows off-line storage of sound data on tape. Other back-panel jacks provide for audio out (mono), headphones, sustain and program-stepping pedals, and MIDI IN/OUT/THRU jacks.

AX-80 Suggested Retail Price $995.00

Cornucopia Price
CAT. NO. 1AAX80F $795 ($55.00 P&H)*

Optional Roland DP-2 accessory (below) will work as either a sustain or program-stepping pedal for the AX-80. For MIDI cables and audio output cable, see page C-12. Sennheiser MS-100 stereo headphones are on page C-13. For powered speakers, see pages C-4 and C-12.

* Dimensions necessitate shipping via air express.



Sir Roland—The Road Warrior

When you need a solid MIDI performer for the road, you can't afford to go unprepared. Pros pack a Roland. JX-8P is its name—real-time control, its fame . . .

AN UNEXPECTED SPARKLE

The JX-8P is both a performer's dream machine and a sound paradox. Because it is both velocity and pressure sensitive, even the slightest change in touch is reflected in the sound. Oscillators' pitch, mix balance, filter cut-off frequency, and amplification level are directly affected by the velocity of key strikes. After-touch pressure on the keys control the vibrato depth, the brilliance, and the volume.

Two oscillators and two envelope generators produce the 6 polyphonic voices. As an analog synthesizer with extensive modulation controls, the JX-8P is expected to produce strong bass color . . . but when it also puts out bright, sparkly bell colors at the top end—like an FM digital synth—musicians shake their heads in utter amazement.

THE GIG MACHINE

The JX-8P offers 64 preset patches—including extremely thick string sounds, cross-modulated metallic sounds, and a variety of special effect sounds. There's on-board storage for another 32 user patches, plus an optional cartridge holding 32 additional programmable patches. An on-stage performer can thus instantly recall 128 patches!

This 25-pound, 5-octave performer also allows you to *name* patches and store the names in memory. When recalled, the fluorescent display spells out the patch name—plus parameter names during editing. You can easily select desired patches and parameters even on a dark stage! Throw in a Patch Chain function for memory recall of up to 8 performance-ordered special effects, and you wind up with a bone-fide Road Warrior . . . one that brings out the best in all connected MIDI instruments.



The PG-800 is a programmer's panel designed exclusively for use with the JX-8P. All synthesizer parameters can be controlled by the PG-800's sliding controls—greatly simplifying and speeding up the creation of new patches, and modifying presets. It neatly fits on top of the JX-8P and is held in place by a magnetic seal. Comes with its own carrying case and connection cable.

Back-panel jacks provide stereo output, MIDI IN/OUT/THRU, and inputs for optional sustain pedal and PG-800 programmer console. Comes with two 1/4-inch to RCA phone cords with 1/4-inch plug adaptors.

JX-8P Suggested Retail Price $1695.00

Cornucopia Price
CAT. NO. 1RJX8P $1249 ($19.00 P&H)

M-16C Cartridge

Cornucopia Price
CAT. NO. 1RM16C $47.95 ($3.00 P&H)

AB-2 Carrying Case

Cornucopia Price
CAT. NO. 1RAB2 $100 ($8.00 P&H)

DP-2 Pedal

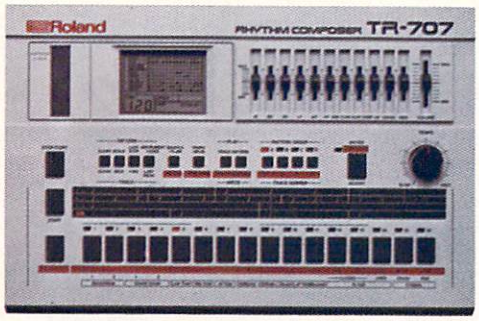
Cornucopia Price
CAT. NO. 1RDP2 $16 ($3.00 P&H)

PG-800 Suggested Retail Price $295.00

Cornucopia Price
CAT. NO. 1RPG800 $225 ($6.00 P&H)

Optional Sennheiser MS-100 headphones are on page C-13. For monitors, speakers and MIDI cables, see pages C-4 and C-12.

Ringo In A Box



Even if you don't have a ticket to ride, you'll be hard-pressed to snare a more versatile drummer . . .

Is it live, or is it a drum machine? With Roland's TR-707 Rhythm Composer, the drums and percussions sounds not only sound real, they are real—digital recordings of real instruments: 2 Bass Drums; 2 Snare Drums; Rim Shot; Low, Mid, and High Toms; Open and Closed Hi-Hats; Ride and Crash Cymbals; Cow Bell, Hand Clap, and Tambourine. Couple this with the machine's dual accentability, variable tempo, stereo output, shuffle and flam capability, 10 individual outputs—allowing tailored processing—plus extensive full MIDI and sync-to-tape features, and you have a world-class drummer that's hard—or is it easy?—to beat at anywhere near this price.

WHERE'S THE BEAT?

The Rhythm Composer's large LCD window provides an unusual feature in drum machines at any price: the ability to see your rhythm notes graphically displayed. The window depicts the pattern in use, tempo in beats-per-minute, MIDI channel operating mode, and the rhythms that each individual instrument plays. This means that you have full control of each measure, with the ability to write or edit down to the 32nd note! This comprehensive display allows even beginners to easily program rhythm patterns that would be difficult to do on other machines.

With The Rhythm Composer's 998-measure memory capacity, you can write complete rhythm tracks (or entire songs), by stringing together up to 64 individual patterns either in "real time" or step-by-step. And—unlike other machines—you can switch between real and step modes even while programming! You can also use the Flam, Shuffle, and Accent functions to give your recordings that "live" feeling.

The TR-707 looks innocent enough, but its bite is actually bigger than its bark! The compact Rhythm Composer weighs a mere 3.3 pounds but packs a punch that matches even the biggest drumset sounds. The unit is completely MIDI-capatible and allows any velocity-sensitive MIDI keyboard to control its sound dynamics. This means that expressive drum playing can be performed directly from a keyboard! And, a tape-sync function allows the TR-707 to synchronize to a multi-track recorder, for building up more complex tracks.

Rhythm programs save to a battery-backed memory; and the unit's built-in tape interface provides data storage on standard cassette tape. Or, you can off-load to an optional memory cartridge (M-64C) with twice (two banks) the internal storage of the TR-707. When plugged into the machine, the cartridge allows free selection between all three memory banks—offering plenty of rhythm variations for both recording and performing.

TR-707 Suggested Retail Price $595.00

Cornucopia Price
CAT. NO. 1RTR707 **$449** ($9.00 P&H)

Options:
M-64C Memory Cartridge
Suggested Retail Price $89.95

Cornucopia Price
CAT. NO. 1RM64C **$79** ($3.00 P&H)

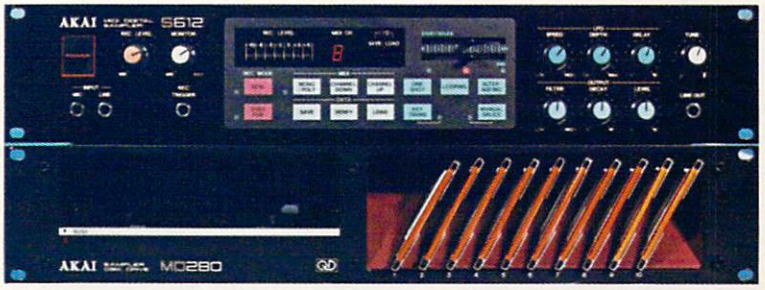
DP-2 Start/Stop Pedal Switch

Cornucopia Price
CAT. NO. 1RDP2 **$16** ($3.00 P&H)

Optional MIDI cables are shown on page C-12. For the Sennheiser MS 100 stereo headphones, see page C-13.

Sonic Alchemy

Magically transform any sound you record into an instrument you can play!



Imagine being able to give your MIDI keyboard virtually any new sound you hear—even your own voice—and be able to do it in a few seconds, with no messy waveforms to fiddle with . . . Envy those Electro-plasmodic space warp sounds your friend gets on his new, holographic laser-drive super synth? Don't envy 'em—just sample 'em! Need a realistic acoustic guitar or piano sound, but don't have the big bucks it takes to equip yourself with what the big boys use? Just sample the real sounds yourself, and play them back with your own modest keyboard.

What does it cost to have all this sonic versatility at your fingertips? Fortunately, Akai has once again made state-of-the-art technology extremely affordable—well within the range of assembling a serious-yet-moderately priced home music studio.

The Akai S612 Digital Sampler uses a 12-bit sampling circuit for a maximum sampling rate of 32K Hz for 1 second, and 4K Hz minimum rate for up to 8 seconds. This provides the flexbililty to meet most needs.

SPECIAL-EFFECT MAGIC

The S612 features a variety of built-in editing functions that will fine-tune—or artistically distort—your sound samples. You can sample any portion of a sound, or use the looping and alternating functions to repeat or alternate sounds; even overdubbing and backwards playback is as simple as a keypress.

When the S612 is connected to a MIDI keyboard, you can play back 6-voice sounds over a 5-octave range. Akai's companion MD-280 Disk Drive uses 2.8-inch disks to store your samples (one per side). A built-in interface on the S612 also allows sample storage dumps to a Commodore Datasette recorder.

S612 Suggested Retail Price $999.95

Cornucopia Price
CAT. NO. 1AS612 **$749** ($13.50 P&H)

MD-280 Suggested Retail Price $299.95

Cornucopia Price
CAT. NO. 1AMD280 **$249** ($8.50 P&H)

10-pack of 2.8-inch disks
Suggested Retail Price $49.95

Cornucopia Price
CAT. NO. 1ABL100 **$42.95** ($3.50 P&H)

NOTE: See page C-12 for the recommended Fostex microphone for sampling.

Bring Your Computer Into The Act

All MIDI software packages require a compatible MIDI hardware interface. These interfaces are either included or are available separately as noted.

Home Computers Make Tracks, Too!

MIDI SEQUENCERS

The MIDI (Musical Instrument Digital Interface) sequencer has become the heart of many electronic musical instrument setups. It has three purposes: 1) to store MIDI performance data as it's played on your MIDI keyboard, 2) to make editing this performance a breeze, 3) to play back your performance (in its original or altered form) by sending MIDI data back to your synthesizer. In short, it's a digital recorder that allows you to "fix" notes before it plays them back!

Editing features are what make MIDI sequencers so valuable to synthesizer players. Imagine being able to change notes, rhythms, synthesizer patch settings, tempos, transposition, and the order of phrases in your compositions—all *after* you perform them. The MIDI sequencer can do all this with the push of a button.

A FORK IN THE ROAD

MIDI sequencers are available in two designs: **dedicated** and **software-based**. Both systems are capable of performing the same basic tasks. But each also has its own advantages—the choice is yours.

DEDICATED—"THE BOX"

Dedicated MIDI sequencers are *stand-alone* boxes that connect to your MIDI keyboard via two MIDI cables. One cable carries performance data from your synth *to* the sequencer, while the other cable carries MIDI data *from* the sequencer to your keyboard. Dedicated sequencers have digital memory, just like home computers. But they can't be used for anything except sequencing—sorry, no checkbook balancing with these units!

Dedicated units are streamlined to do their job efficiently. Their controls usually resemble those on tape recorders. These units are also generally compact and portable, making interface to your keyboards a snap.

Most dedicated sequencers allow you to store sequence data on cassette tape. Some allow storage on optional memory cartridges, and other more expensive units have built-in disk drives for storage on disks.

SOFTWARE-BASED

The software-based sequencer is designed to take advantage of the power of your home computer, while performing the same functions of dedicated sequencers. These sequencers require a specific amount of computer memory, a monitor, a disk drive, and a MIDI interface that is compatable with your particular system.

This cost-effective sequencer design offers many advantages over dedicated units—especially if you already own a computer. The advantages include visual display of recorded tracks, visually assisted editing, and disk storage of sequences. And because the sequencer functions themselves *reside* on disk, these programs are periodically updated by the manufacturer.

THE BEST?

Your particular needs will determine whether you purchase a dedicated sequencer or a software-based system. If it's portability and simplicity you want, explore the dedicated models. If you already own a home computer and anticipate a continually growing MIDI setup, look at the software-based systems.

Today the electronic music market is flooded with a myriad of sequencer systems. But the software-based programs on this page, and the dedicated model on page C-4 represent the finest value on the market today!



Apple II+, IIe: Passport's **MIDI 4+** (4 tracks) or **MIDI 8+** (8 tracks) software packages require Passport interface **MH-01A** or **MH-02A** (which includes tape sync). These packages give you unlimited overdubbing, punch in/out for easy editing, single-step playback, fast forward/rewind, auto-correct to 32nd-note triplets, real-time tempo change, and drum sync. Single-step editing makes for easy selection of where changes are to begin.

Software includes: one 5-1/4" disk and complete instruction manual.

Interface includes: two MIDI cables and manual.

Requirements: 48K, disk drive, monitor

Recommended accessory: RCA to RCA cable for tape sync with MH-02A (page C-12).

		Suggested Retail Price	Cornucopia Price
CAT. NO. 1P4A	MIDI 4+	$129.95	$109 ($4.00 P&H)
CAT. NO. 1P8A	MIDI 8+	$169.95	$149 ($4.00 P&H)
CAT. NO. 1PMH01A	MH-01A Interface	$149.95	$129 ($4.00 P&H)
CAT. NO. 1PMH02A	MH-02A Interface (w/tape-sync feature)	$199.95	$169 ($4.00 P&H)



Commodore-64: The **MH-01C** interface plugs into the C-64 user port and works with either of Passport's **MIDI 4+** (4 tracks) or **MIDI 8+** (8 tracks) software packages. The software features a clearly designed display, unlimited overdubbing, punch in/out for easy editing, single-step playback, fast forward/rewind, auto-correct to 32nd-note triplets, real-time tempo change, and drum sync. Single-step playback allows you to step through 24 increments of each beat to pinpoint changes.

Requirements: disk drive, monitor.

Software includes: one 5-1/4" disk and complete instruction manual.

MH-01C interface includes: three MIDI connections and manual + 2 MIDI cables.

		Suggested Retail Price	Cornucopia Price
CAT. NO. 1P4C	MIDI 4+	$99.95	$89 ($4.00 P&H)
CAT. NO. 1P8C	MIDI 8+	$149.95	$129 ($4.00 P&H)
CAT. NO. 1PMH01C	MH-01C Interface	$129.95	$109 ($4.00 P&H)



Atari: Hybrid Arts **MIDITRACK II** sequencer gives you 16 MIDI recording tracks, punch in/out capability, step editing, rhythmic auto-correct, track naming, and the most complete sync options on the market. This package comes complete with the required **MIDIMATE** interface and two MIDI cables.

Requirements: Atari 800, 800XL, 130XE, 400, or 1200XL (48K min), Atari 810 or 1050 disk drive, and monitor.

Package includes: 1- 5 1/4" disk, manual, interface and 2 MIDI cables.

		Suggested Retail Price	Cornucopia Price
CAT. NO. 1HA2	MIDITRACK II (with MIDIMATE)	$349.00	$279 ($6.00 P&H)



Macintosh: Opcode's **MIDIMAC** sequencer & interface take full advantage of Macintosh graphics capabilities. This sequencer mimics the controls of a standard tape recorder on screen! It records up to 24,000 notes, on 26 sequences with 10 tracks each. You can even play back 16 sequences at once!

Requirements: Macintosh (512K).

Software includes: 1-3 1/2" disk, manual

Interface includes: hardware module which plugs into the modem or printer port.

Recommended Accessories: 2 MIDI cables (page C-12).

		Suggested Retail Price	Cornucopia Price
CAT. NO. 10S1	MIDIMAC Interface	$125.00	$109 ($4.00 P&H)
CAT. NO. 10S2	MIDIMAC Sequencer	$150.00	$129 ($4.00 P&H)



IBM PC: The Roland **MIF-IPC** interface kit and **MPU-401** (MIDI processing unit) along with Cherry Lane's **Texture** turn your IBM PC into an 8-track sequencer. **Texture** uses a unique "modular recording" process to combine up to 64 "patterns" into completed compositions—allowing you to compose your tune in musical phrases and let the program link them together.

Requirements: 128K, disk drive, monitor.

Software includes: 5 1/4" disk and manual.

Interface includes: MIF-IPC interface kit + cable to the MPU-401. MPU-401 comes with a manual.

Recommended accessory: 2 MIDI cables for MPU-401 (page C-12).

		Suggested Retail Price	Cornucopia Price
CAT. NO. 1CLT1	Texture	$199.00	$169 ($4.00 P&H)
CAT. NO. 1RM1F1PC	MIF-IPC Int. Kit	$100.00	$89 ($4.00 P&H)
CAT. NO. 1RMPU401	MPU-401 Proc. Unit	$200.00	$169 ($5.00 P&H)



TI-99/4A: The **TEX-SEQ** interface cable and software from World Class Software allows for note-by-note entry of musical compositions, and playback on any MIDI synthesizer. Software provides 4 play tracks and supports several different time signatures. Files may be saved to disk for easy storage and recall.

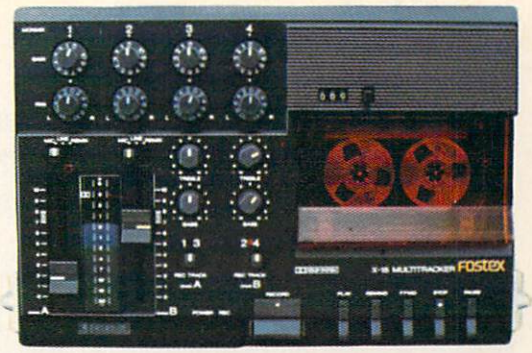
Requirements: Extended Basic, 32K memory expansion, disk drive, and monitor.

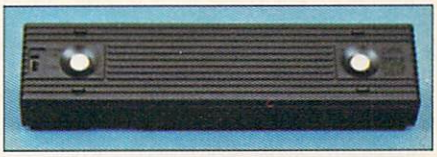
Package includes: MIDI interface cable, one 5-1/4" disk, and manual.

		Cornucopia Price
CAT. NO. 1WCT1	TEX-SEQ	$49.95 ($4.00 P&H)

Studio To Go

For a take-out order that's musically nutritious, high in unsulfured creativity, low in weight, and that won't upset your budget . . .





With the attachable battery pack, the X-15 Mini Studio is ready to go wherever your inspiration leads.

Look no farther for a full-featured mini recording studio with maxi capabilities—as easy to use on the go, as it is at home! The Fostex X-15 puts multi-track recording technology at your fingertips. With it, you can create great sounding demo tapes—just like the pros do—with built-in features that let your creativity run wild . . .

The X-15 is a complete, ultra-compact 4-track cassette mixer/recorder system. It allows you to build your music one track at a time or record in true stereo; listen to previously recorded parts while overdubbing others; fix mistakes anywhere on a track with punch-in/out capability, ping-pong (i.e., combining tracks 1, 2, and 3 and mixing them onto track 4—thus freeing them up for overdubbing new parts); and mix down four tracks in stereo (or mono) onto another recorder.

The built-in two-channel input mixer has linear slide faders with LED peak-reading bar graph meters allowing careful control and monitoring of levels. Use the two front-mounted 1/4" phone jacks to input signals from either mics or instruments, or a pair of RCA input jacks for line level signals. Each input channel offers shelving-type equalization with ± 12 db cut or boost at both 100 Hz and 10 KHz. Track selectors let you record up to two tracks at a time with Dolby B noise reduction. A separate, 4-channel monitor and mixer boasts individual gain and stereo pan controls for each tape track.

Other features include soft-touch transport controls with a cueing function in the fast wind modes, a headphone jack with level control, and a pitch control (± 15%) which allows you to tune the X-15 to instruments, other tapes, or generate special effects in record and playback modes.

If desired, you can tap the output signals directly off the playback head (bypassing the internal mixer) using the four direct-out RCA jacks on the side of the machine. This allows you to transfer your basic tracks to a larger multi-track recorder or mixer with maximum fidelity. The X-15 has excellent frequency response and signal-to-noise ratio to faithfully capture and play back your most demanding musical creations.

THE GO-ANYWHERE MACHINE

The X-15 is a cinch to carry. With battery pack (and batteries) included, the unit is about the size of an average dictionary, and weighs less than seven pounds! It's the perfect tool for keeping a musical diary. Take it along to capture that idyllic music-making session in the park. Keep it beside you in your car or van—so that when something on the radio triggers an idea, you can pull off the road, hum the lick, sing the vocal, or make a note to yourself about a synth sound or drum beat. Can't sleep because some strange melody is haunting your thoughts? Hook up your keyboard to the X-15 and unleash your ideas note-by-note, track-by-track . . . This musician's dream machine is always ready when you're really cookin'.

AT HOME IN A MIDI STUDIO

The X-15 really shines in your home electronic-music studio. You can sequence your MIDI keyboard synth and drum machine, and record them on tape. Then plug in an electric guitar and a microphone for capturing some live instrument and vocal tracks. Finally, blend it all together with stereo imaging to create that full, rich, 3-D sound you hear on tapes by your favorite groups. After the session, unwind by listening to these same commercial tapes on the X-15. Because this unit operates at the standard cassette 1 7/8 ips speed, it doubles as a super-quality stereo tape deck. Dub tapes from your CD or stereo system using the pair of RCA line inputs. You can even add your own ideas to these recordings, using the left-over two tracks! Put one of these dream machines into the hands of a musician—budding or pro—and watch the creative sparks fly.

Fostex also makes an optional accessory for the X-15, the MN-50 Mixer/Compressor, which greatly enhances the recording process. This single output mixer features four line inputs (normally used for tape-in), plus an additional selectable mic/line input, and a built-in variable compressor. The MN-50 is especially useful for merging tracks (ping-ponging) with the X-15. As tracks are bounced, additional parts may be added simultaneously—thereby expanding the number of parts in the sub-mixes. Up to 10 tracks worth of material can be captured using this technique, never exceeding second-generation recordings. You'll find the compressor useful for adding punch to vocals and smoothing out levels from percussion instruments. This palm-sized unit comes equipped for battery or optional AC adaptor operation.

We want to make multi-track home recording as accessible and affordable as possible. So, as a service to you, Cornucopia has put together a Fostex Action-Pack of accessories that, when purchased together with the X-15, provides a special savings bonus.

X-15 Suggested Retail Price $495.00

Cornucopia Price
CAT. NO. 1FX15 **$349** ($7.00 P&H)

X-15 with Action-Pack Accessories
Suggested Retail Price $703.95

Cornucopia Price
CAT. NO. 1FX15C **$475** ($12.00 P&H)



MN-50 Mixer/Compressor	$55.00
(2) Punch In/Out Pedals	$24.00
AC Adaptor	$30.00
X-15 Video Instruction Tape	$39.95
M-3 Microphone	$60.00
Item-By-Item Total	**$208.95**

Fostex Action-Pack

The **Fostex Action Pack** offers several indispensable items for the active multi-tracker. First, the Pack includes a concise VHS video instruction cassette that demonstrates a variety of helpful recording techniques with the X-15. Second, it includes a heavy-duty power supply that saves batteries during those long sessions at home. Two punch in/out pedals facilitate hands-free recording when you are both recording engineer and performer. With the included MN-50 Mixer/Compressor, you get a high-quality Fostex M-3 microphone. Bundled with the X-15 (see price above), The Action Pack saves you almost $230!

THE 4-TRACK RECORDING THAT CHANGED IT ALL

The release of *Sergeant Pepper's Lonely Hearts Club Band*, the watershed album by the Beatles and producer George Martin, marked the beginning of a new era in popular music. The boldly innovative music spoke directly to a whole generation of people worldwide. Especially significant is how this music came into being. The Beatles, under the creative direction of George Martin, realized that the then-new technology of multi-track recording made fresh and different approaches to music creation possible. Music could now be played and recorded in bits and pieces, over time—with complete syncronization—to be assembled (or "mixed") into a finished work at a later date. Before Sgt. Pepper, most records were made by bringing the artist(s) into the studio, and capturing their live "performance" on tape. The Beatles turned the studio *itself* into an instrument. It was the 4-track recorder that allowed them to do this. Tracks were recorded, mixed, bounced, and blended in creative and unusual ways. George Martin even went so far as to cut tape apart, randomly resplicing it to create new "instruments" out of recordings of traditional ones. Also, the Beatles were able to play several instruments simultaneously in this recording, thanks to the magic of multi-track.

In short, Sgt. Pepper turned the world on its ear, introducing the concept of an "audio canvas" to an appreciative, music-loving public.

While the impact of their work remains undiminished (Sgt. Pepper is arguably the greatest pop music achievement of our time), some things have changed. With the introduction of the X-15, a 4-track recorder and mixing console—which used to weigh hundreds of pounds, take up the space of a washer and dryer, and cost more than an Italian sports car—is now available in a single, portable, inexpensive unit. Take advantage of this new tool. Who knows? The world's next 4-track masterpiece may be your own . . .

"Con The Pros!"

Can a new breed of 4-track "mini-studio" take on the far-more-costly 8-track standard? In many applications, you'll be hard-pressed to tell the reel score . . .



Until recently, serious musicians and recording engineers had to spend many thousands of dollars to equip their home studio with an 8-track recorder and mixing board to get that professional-sounding "big mix" for their all-important demo tapes . . .

Now, with the introduction of the Audio Technica AT-RMX64, the world of convenient and affordable 4-track cassette mastering takes on a new dimension—with equipment specs and features that were formerly unheard of in this type of "mixer/cassette" unit.

SOUND FLEXIBILITY

With 6 input channels, you can mix instruments, microphones, and other sources together, record the mix on up to 4 tracks, while sending to a house PA. And with a choice of either Dolby B and C noise reduction and punch-in/punch-out availability on any track, you are guaranteed superior sound quality every time you record. There's even *two* recording/playback speeds—3 3/4 ips for best frequency response (20 to 18,000 Hz with 3 dB variance from 40 to 15,000 Hz) and lowest wow and flutter (.04% RMS); and 1 7/8 ips for playback and recording of standard-speed cassettes. The electro-mechanical design of 3 direct-drive DC motors cushioned by a chassis of nearly 50 pounds provides rock-steady recording results.

PROFESSIONAL FEATURES

Each of the 6 input channels accept balanced or unbalanced mic or line signals with patch in/out for every channel. A total 60dB of available input attenuation guards against overloading. Professional features include a solo switch that lets you monitor any individual input or return. The AT-RMX64 operates with a studio-quality signal strength of +4 dbm instead of −10 dbm as most mini-recorders do. You can even vary the recording speed up to 15% to match any instrument pitch. Two-band parametric EQ is available in each input with a choice between peak/dip control shelving and continuously variable frequency in gain controls. Two auxiliary sends are switchable from pre-EQ and fader to post-EQ and fader. You can assign all inputs and returns to any of the 4 available sub buses. With powerful features like these, the mixer section can compete in versatility with many higher priced 8-channel boards!

GROW WITH IT

The AT-RMX64 is a well-engineered unit that will grow with you as your musical and recording skills mature. The machine is compatible with both home and professional cassettes; its outputs meet all professinal standards both as a recorder and a live mixer. If you're looking for professional recording features in a turbo-charged "mini-studio," the AT-RMX64 gives you all you need and more.

AT-RMX64 Suggested Retail Price $1595.00

Cornucopia Price
CAT. NO. 1ATRMX64 $1259 ($30.00 P&H)

Optional Fostex powered monitors and Fostex M-3 microphone are shown on page C-12. For World Class Audio™ high-bias tapes, see below.

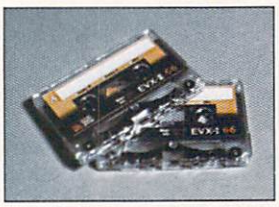
Rosemary's Revenge

If a certain past President had been able to specify 2 World Class Audio C-99 tapes, his personal secretary might not have lost those crucial 18 minutes . . .

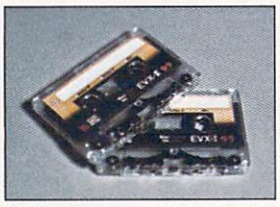
What Makes A World Class Tape?

Could it be the extra length?

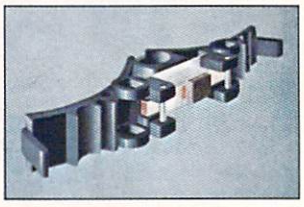
World Class Audio™ doesn't manufacture C-60 or C-90 cassette tapes like everyone else. Theirs are C-66 and C-99, giving you 6 or 9 extra minutes of what can sometimes be crucial recording time. The 66-minute tape is ideal for recording material from Compact Disks (CDs) and classical LPs having longer program duration. The 99-minute tape is perfect for recording 2 shorter CDs, LPs, or cassettes on 1 tape. You'll never again "come up short" and miss capturing that last full song on each album.



C-66 cassettes are ideal for recording material from CDs and classical LPs with longer program duration.



C-99 cassettes are perfect for recording 2 shorter CDs, LPs, or cassettes on 1 tape.



The unique "bridge" insert forms a "unipath" guide that maintains superior head azimuth alignment.

Could it be a "Shell of a Deal?"

World Class Audio cassette shells have been specially engineered and built in the USA to overcome the head azimuth problems that plague all other shells on the consumer blank-tape market today. First, a special "UniPath" bridge piece (see photo) does away with the problem of mis-matched shell halves forming the tape path. Second, the uneven screw-torque and spot-fastening of the shell halves has been eliminated by sonic-welding of the entire shell seam. This added stability assures consistent and reliable performance. And with the inclusion of polyester "creased-sheets" on either side of the tape, even packing and tape flow are assured.

Could it be a bias toward perfection?

Choosing the perfect tape for an application is a matter of bias. World Class Audio's premium-quality normal-bias tape (120 μS EQ) provides excellent frequency response and dynamic range for a wide variety of tape recording needs. For cassette machines with high-bias (chrome) settings, World Class Audio's very fine chromium-oxide tape (70 μS EQ) may be your preference. High-bias tape is recommended for dubbing Compact Disks or other demanding source material, and is specified by virtually every manufacturer of multi-track cassette machines.

Could it be a "World-Safe" warrantee?

What if the cassette tape fails? These World Class Audio cassettes are meticulously designed *not* to fail, but in the unlikely event one does—for any reason whatsoever—return it to us for a brand new replacement. Even if you are paddling down the Amazon River, recording exotic jungle sounds when your sturdy cassette fortuitously intercepts a poison dart—send it in and get a new one!

Or, could it be our remarkable prices?

What is the price for all this World-Class quality, peace of mind, and headache-saving extra length? Just take a look at these remarkable values:

C-66 EVX-1 normal-bias 120μS EQ
Cornucopia 10-pack price **$22.95**
CAT. NO. 1C66N10 ($2.50 P&H)

C-99 EVX-1 normal-bias 120μS EQ
Cornucopia 10-pack price **$25.95**
CAT. NO. 1C99N10 ($2.50 P&H)

C-66 EVX-2 high-bias 70μS EQ
Cornucopia 10-pack price **$26.95**
CAT. NO. 1C66H10 ($2.50 P&H)

C-99 EVX-2 high-bias 70μS EQ
Cornucopia 10-pack price **$29.95**
CAT. NO. 1C99H10 ($2.50 P&H)



It's not inquisitive like R2D2, and not as chatty as C3PO, but watch how it blasts away hiss, translates your orders, and clones your tapes . . .

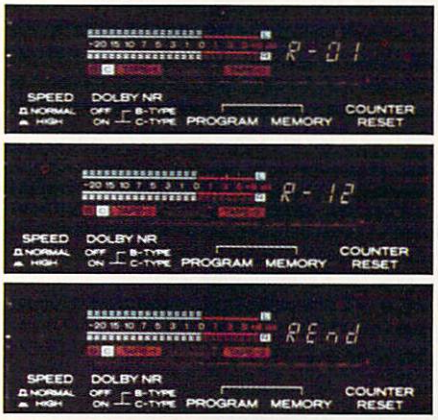
Dual Dubbing Droid

The Hitachi D-W800 won't make the coffee or put out the kitty, but for computer-controlled editing of musical selections, sound-on-sound mixing, and high-speed dubbing, it's the cream of the crop . . . the cat's meow.

CREATIVE CYBERNETIC CONTROL

Copying tapes can be time consuming, especially if you have to search for the song you want first. It can take even longer if you want the songs recorded in a certain order (such as songs 2, 3, 5, 7, and 8, or maybe even 6, 3, 8, 2, and 5). This becomes mere child's play with the D-W800. Its microprocessor allows you to play and record *any* songs in *any* order you want—up to 15 selections under full computer control.

After you've programmed in your selections, the microprocessor precisely takes up the leader tape to prevent losing the beginning notes of your recordings. While one deck plays back your selections, the other records them—at *twice* normal speed, if you so specify. Then the record deck stops and adds 4 seconds of silence while the play deck either fast forwards or rewinds to find the next song. When everything is in sync, the four motors and smart electronics of both decks continue the dubbing process. There's virtually no need for you to intervene and risk flubbing a dubbing!



The deck's clear display guides you step-by-step through the process of programming the high-speed edit and recording order for your musical preference. Then the machine takes over the work!

SIT BACK AND RELAX

D-W800 also makes *listening* to music effortless. The Long Play mode lets you play two tapes sequentially, up to 8 times each. This unique feature provides hours of music pleasure without getting up to change cassettes. The random program and search features let you automatically find the beginning of any song, jump forward to the next song, or program the machine to repeat songs over and over.

There's even a timer function that allows the machine to power up in either record or playback mode (under control of an accessory audio timer—see page C-12) to capture that special concert broadcast while you're away, or awaken you to your favorite Vivaldi piece.

JOIN IN THE FUN

The D-W800 is an ideal cassette deck for singers. Plug in a microphone and sing along with music playing on one deck, while recording everything on the second deck. You can also dub over albums or radio broadcasts using the machine's line input.

SUPERIOR SOUND COMES STANDARD

Its true that feature-for-feature no other dubbing deck in this price range even comes close to matching this Hitachi machine, but all this design sophistication would be for naught if the deck didn't provide clean sound. No worry here. The D-W800's wide frequency response and both Dolby B and C noise reduction provide superior results. The record bias circuitry automatically senses and adjusts for all normal, chrome, and metal tapes.

With features like these, its easy to see why the D-W800 is an excellent choice for all your taping needs.

D-W800 Suggested Retail Price $389.00

Cornucopia Price
CAT. NO. 1HDW800 $229 ($12.00 P&H)

Optional Sennheiser MS 100 stereo headphones are shown on page C-13.

SPECIAL GIFT

To start you off on the right foot, Cornucopia will send all purchasers of the D-W800 a FREE World Class EVXII C-99 High-Bias audio cassette tape. This is our gift to you even if you decide to return the tape recorder. We want the *first* tape you try in this unit to be the best available, so that you'll discover just how good a Dual Dubbing Droid can be.

CASSETTE TAPE TALK

WHAT IS BIAS?

Bias is a high-frequency signal that the recorder applies to the recording head along with the audio signal to improve the frequency response of the recording. Each tape formulation has an optimum bias level.

HEAD AZIMUTH ALIGNMENT

Head azimuth alignment refers to the degree of perpendicularity between the magnetic gap of the tape head and the magnetized track of information on the tape. Because cassette tape moves slowly, the recorded wavelengths of high-frequency signals on the tape are very short (2.4 microns for a 20-kHz signal). In order to resolve these tiny wavelengths, the tape head must have an incredibly narrow gap (under 2 microns), and must have accurate azimuth alignment.

If your tape head is out of alignment and you play a tape recorded with Dolby noise reduction, it will sound distant and muddy because of lost high-frequency response and also because of Dolby decoding errors caused by this loss.

The physical tolerances of the cassette tape shell itself compound this problem. Because the pressure pad, spring, and tape guide rollers are held in place by the two halves on most cassette shells, it is the shell that plays the principal role in maintaining tape-to-head contact and alignment. Variations in the shape of the shell and even in the torque of the screws that hold it together can affect this critical head-azimuth alignment.

MIXING

Mixing is the process of combining two or more audio signals (from microphones, tape decks, musical instruments, or signal processors) into a single new signal with the aid of a mixer or mixing console. The mixed signal is then usually recorded on tape to preserve the mix.

NOISE-REDUCTION SYSTEMS

Dolby B—the most common noise-reduction system for cassette recording—provides up to 10 dB of high-frequency noise reduction. The newer Dolby C system provides improved performance and up to 20 dB of noise reduction.

Another popular noise-reduction system for consumer applications is "dbx Type II," which is capable of 30 dB or more of broadband noise reduction.

Both Dolby and dbx noise-reduction systems work on the "encode-decode" principle. The noise-reduction system processes or encodes the signal during recording, and decodes it during playback. The Dolby systems are sliding-band companders, while the dbx system is a linear compander.

A "compander," or compressor-expander, compresses the dynamic range of the audio signal during the recording process, giving the signal a more uniform level to ride over the noise. It then expands the dynamic range during playback, restoring the variations between loud and soft. Sliding band companders are sensitive to a varying portion of the audio-frequency spectrum. Dolby B affects only the high frequencies, and Dolby C affects only the mid and high frequencies. Linear companders like the dbx system, by contrast, affect the entire audio frequency spectrum.

OVERDUBBING

The process of recording tracks at different times on the same multi-track master tape (analog or digital) is referred to as overdubbing.

SIGNAL-TO-NOISE RATIO

The signal-to-noise ratio (S/N) is the difference in dB between the amplitudes of the desired signal and the noise that exists under no-signal conditions.

TAPE EQUALIZATION

Tape recorders boost high frequencies according to a "record equalization curve" during recording, and reduce them according to a "playback equalization curve" on playback. Record equalization causes the signal to ride over high frequency tape noise more effectively. Playback equalization reduces high frequencies and noise, so that the high-frequency components of the signal are returned to their previous levels with reduced noise. Normal Bias tapes use 120 μS playback equalization. High Bias (including the premium Chrome-type) tapes use 70 μS playback equalization.

TRACK BOUNCING

The process of mixing together two or more tracks of a multi-track master tape and simultaneously recording them on a different track of the same tape is called "bouncing."

TRACK FORMATS

A track is a magnetized region containing recorded information that runs lengthwise along an audio tape. Cassette recorders conform to one of 3 track formats: 2-track monaural, 4-track stereo, and 4-track multi-track.

Monaural tapes are divided in the middle, with one track for each half of the tape. The active portion of the record or play head sits above the first track. When the tape is turned over, the second track will be under the head gap.

The 4-track stereo cassette format uses two tracks side-by-side in each direction. Each head has two gaps that "see" the pair of tracks on one half of the tape. Turning the tape over allows the head to sense the other pair of tracks. Because there are two tracks on each half, each track takes up a quarter of the tape width.

The 4-track multi-track format records on all four tracks only in one direction and employs heads with four gaps. This format is used to produce four-track multichannel recordings.

WOW AND FLUTTER

An expression of the degree of steadiness of the tape speed. Slow variations are called wow, fast variations are called flutter.

Fostex M-3 Microphone

Complete your sound system with this uni-directional microphone. Perfect for vocals as well as instruments. 90-16,000 Hz.



M-3 Suggested Retail Price $60.00

Cornucopia Price **$49.95**
CAT. NO. 1FM3 ($3.50 P&H)

Boss BX-600 Mixer

Combine outputs of up to 6 electronic instruments with the Boss BX-600. This 6 IN/2 OUT audio mixer is tailor-made for hooking up several synths and a drum machine to your home stereo system or to a set of powered monitors. Enjoy the convenience of individual volume and pan controls for each input. This high-quality mixer is a perfect match for any instruments in this catolog.



BX-600 Suggested Retail Price $225.00

Cornucopia Price **$179**
CAT. NO. 1RBX600 ($6.00 P&H)

Korg KMT-60 MIDI-Thru Box

Connect up to 6 MIDI instruments to a single MIDI sequencer with this flexible device. The KMT-60 MIDI- Thru Box eliminates "MIDI delay" that results when you use MIDI THRU jacks to daisy chain several MIDI instruments together. With the KMT-60, each instrument receives the same MIDI data from your sequencer—at the same time.



KMT-60 Suggested Retail Price $69.95

Cornucopia Price
CAT. NO. 1KKMT60 **$59.95** ($4.00 P&H)

Korg Equipment Stands

The Korg multi-tier equipment stand provides a solid foundation for your keyboards, drum machine, sequencers, etc. Featuring lightweight aluminum construction, this attractive stand is at home on the road and in your studio. It comes complete with a built-in microphone boom, carrying bag, and two support arms per tier. The stand is available in chrome with either 2 or 3 tiers. Order extra tiers and support arms below.



2-Tier Stand Suggested Retail Price $240.00

Cornucopia Price **$199**
CAT. NO. 1K2TKS ($15.00 P&H)

3-Tier Stand Suggested Retail Price $280.00

Cornucopia Price **$229**
CAT. NO. 1K3TKS ($18.00 P&H)

Additional Tier (without support arms)

Cornucopia Price **$49.95**
CAT. NO. 1KTRC ($6.00 P&H)

Additional Pair of Support Arms

Cornucopia Price **$24.00**
CAT. NO. 1KKSR ($3.00 P&H)



Powered Monitors

Highly versatile, these self-powered monitors accept line feeds from instruments, pre-amps, and recorders. Use the Fostex Personal Monitor for instrument monitoring, studio reference, remote recording, video production, and A/V presentations. Unbelievable sound quality from so small a box.

Fostex Personal Monitor
Suggested Retail Price $149.00

Cornucopia Price **$124.95**
CAT. NO. 1F6301 ($6.00 P&H)

Accessory Cables

10-Foot Multi-Purpose Patch Cable with 1/4" Phone Plug to RCA Plug for connecting synthesizer and drum machine OUTs to tape recorder LINE INs. Useful for connecting tape recorders to powered monitors. Also used for transferring patch data from Akai AX-80 to a high-quality stereo cassette deck. Comes with RCA to 1/4" adaptor for connecting synthesizers, drum machines, and guitars to stage amps, effects devices, mixers, and powered monitors. This cable meets virtually all studio needs.

Cornucopia Price **$10.00**
CAT. NO. 1RPJ1 ($2.00 P&H)

RCA to RCA Stereo Cables for connecting audio components. Available in three lengths.

Cornucopia Price		($2.00 ea P&H)
1 meter	CAT. NO. 1HCRA201	**$3.50**
2 meter	CAT. NO. 1HCRA202	**$4.00**
3 meter	CAT. NO. 1HCRA203	**$5.00**

MIDI Cables: MIDI connection cables for linking synthesizers, sequencers, drum machines, computers, and MIDI-effects devices.

Cornucopia Price		($2.00 ea P&H)
2.5-meter	CAT. NO. 1RMSC25	**$12**
5.0-meter	CAT. NO. 1RMSC50	**$16**

Multi-Cable Packs

Versi-Pack (2 cords) . . . . . **Save 12%**
(2) 10-foot Multi-Purpose Patch Cables

Cornucopia Price **$18**
CAT. NO. 1WCVP1 ($3.00 P&H)

Audio Pack (4 cords) . . . . . **Save Over 32%**
(1) 1-meter RCA-RCA
(2) 2-meter RCA-RCA
(1) 3-meter RCA-RCA

Cornucopia Price **$13.90**
CAT. NO. 1WCAP1 ($4.00 P&H)

Super Studio Pack (7 cords) . . . . . **Save Almost $18.00!**
(2) 10-foot Multi-Purpose
(1) 1-meter RCA-RCA
(1) 2-meter RCA-RCA
(1) 3-meter RCA-RCA
(2) 2.5-meter MIDI

Cornucopia Price **$48**
CAT. NO. 1WCSP1 ($5.00 P&H)

MIDI—WHAT IS IT? WHAT DOES IT DO?

A few years ago, a wave of excitement hit the electronic music industry. After months of work, several competing synthesizer manufacturers *agreed* on a system that enabled synthesizers, drum machines, sequencers, effects devices, and home computers to communicate through a standard interface. The implications were massive. The response was universal. And best of all, the system worked! This significant development was named: **MIDI** (Musicial Instrument Digital Interface).

THE "INS" AND "OUTS" OF MIDI

The MIDI system operates on one very simple premise—a digital command is either *transmitted* or *received*. MIDI instruments usually have at least one MIDI-IN and one MIDI-OUT jack to send and receive MIDI messages. In addition, some MIDI instruments offer a MIDI-THRU jack that outputs the same MIDI message that is received by the IN jack—a handy feature for sending the same message to two instruments.

MIDI messages travel from the transmitter's OUT to the receiver's IN via a special cable called a MIDI cable. These cables have a 5-pin DIN plug on each end that is designed to plug into the 5-pin MIDI jacks on your instrument.

THE MIDI MESSAGE

MIDI is a "communication system" and therefore has its own language. MIDI's particular language happens to be digital. This means that it has only two letters in its alphabet (1 and Ø). All its "words" are spelled with some combination of these two characters (e.g., ØØØØ1Ø1Ø or Ø111ØØ11). These numbers are created electronically and are transmitted one at a time (serially) at 31.25 Kbaud—very fast! This whole process resembles the data communication system used in the computer industry.

MIDI transmits messages that can include the following: the note you play, when you played it, the velocity of your keystroke, how much aftertouch you used, any program (patch) changes you made during your performance, pitch bend and modulation wheel changes, timing information (used to synchronize several instruments), and system exclusive information (often used to send patch information to "patch libraries" or to transfer information between two identical instruments).

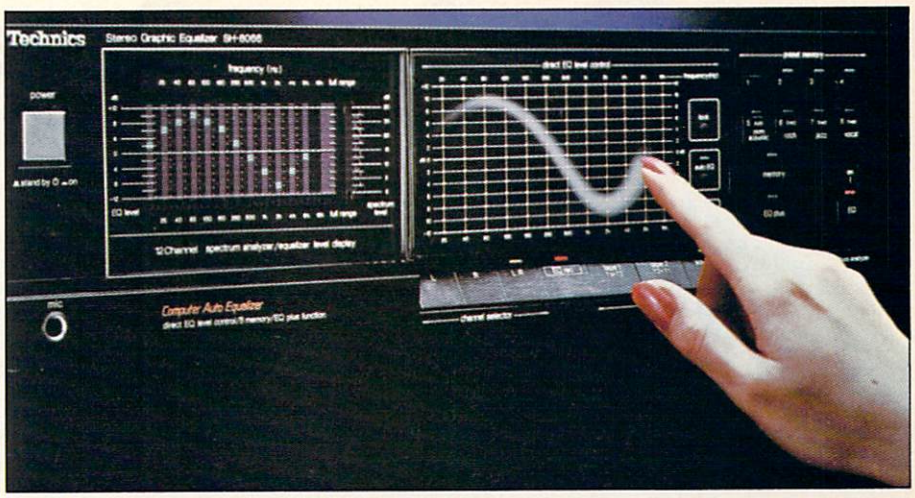
Don't worry! You don't have to understand the MIDI language to take advantage of its capabilities. There are countless products available that make it easy to use.

GETTING MIDI TO WORK FOR YOU

MIDI is designed to allow you, the musician, more freedom to control and change your instrument setup. This means that instruments equipped with MIDI will not become obsolete after only a few years of use, like so many other musical instruments. Any instrument with MIDI jacks will remain useful as part of your studio, because it can always be connected to newer equipment as you expand.

There are many ways to create and interconnect MIDI studios. Just remember that MIDI messages must "flow" *from* one instrument *to* another (always connect your OUTs or THRUs to your INs), and you'll be well on your way to enjoying the amazing benefits of MIDI electronic instruments.

Can't Get No Satisfaction? You Need— The Equalizer!



The new audio era with its Compact Discs, audio-video sources, home music studios, and other fascinating dimensions needs a new kind of tool. Enter the Technics SH-8066—a graphic equalizer so advanced that it sports its own internal computer for automatic equalization—completely eliminating the hassle of trying to find the best slider settings for flat response. This personalized "room acoustics" curve is placed in memory (as one of the unit's 8 preset curves) to compensate for imperfections in your listening room.

When you're backed into a corner . . . with no place to turn . . . and frequencies are stacked against you—you need help FAST! But one finger is all it takes to bring instant action. No sonic problem is too big for —The Equalizer!

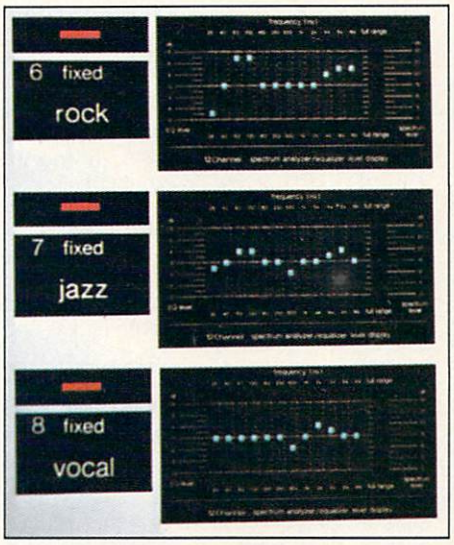
WORLD'S FIRST DIRECT-TOUCH EQ

With the SH-8066, you don't have to fiddle with a long string of adjustible sliders. Instead, a direct-touch control panel lets you "draw" with your finger the kind of equalization curve you desire. Simply slide your finger across the touch-sensitive control panel, and watch the equalization curve appear on the fluorescent display of the built-in spectrum analyzer. Then press one button to lock this curve into place.

HAVE IT YOUR WAY

You can choose from 8 *preset* equalization curves—3 factory presets (rock, jazz, vocal), 4 of your own curves, and the "room acoustics" curve calculated by the equalizer's internal computer, using its built-in pink noise generator and supplied test microphone. The unit provides 12dB boost or cut (in 2dB steps) at each of 12 frequency bands.

But which curve should you pick? If you can't decide, select different curves by pressing the appropriate button and compare the unique sound of each mixture. Or, with the "EQ Plus" function, you can stack any two curves to get that perfect equalization. Try stacking the room acoustics curve with the vocal curve to create the equalization curve of your favorite singer.



Any two of the curves, like those shown above, can be "stacked" for perfect equalization.

If you enjoy watching the dynamic levels of a particular song, check out the colorful bar graph that lights up on the spectrum analyzer display for a fascinating reflection of the changing energy content in the audio spectrum. Or press one more button and the SH-8066 depicts the EQ curve as a series of glowing dots.

JOGGERS & DRIVERS REJOICE

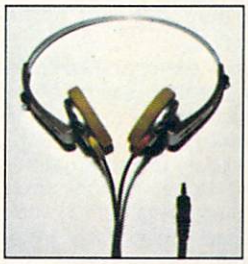
For those who listen to music all day long—while driving, jogging, or working—the SH-8066 has another ingenius feature: it lets you tape music that is equalized to any of your particular specifications. You can, for example, record music that is specially equalized to fit the characteristics of your car stereo or portable headphone cassette player. And, with the unit's two-channel (12 bands per channel) EQ circuitry, you gain versatile control over two tape decks. The SH-8066 helps you get the most from *all* of your cassette-playing stereo systems!

SH-8066 Suggested Retail Price $500.00

Cornucopia Price
CAT. NO. 1TSH8066 $369 ($12.00 P&H)

Sennheiser Switch-Hitter

Highest-quality headphones for go-anywhere music pleasure



What could be better than a single set of headphones for your CD player, cassette deck, synthesizer, drum machine, and personal portable? **The Sennheiser MS 100**'s new design in headphones brings the most out of any audio product, and the best of any synthesizer—a very demanding test, indeed. With a frequency range of 20 Hz to 20 KHz, the **MS 100** headphones are the perfect match for both the Technics SL-P3 CD player, and the Hitachi D-W800 Dual Dubbing Deck. "Open-Aire" technology permits total listening enjoyment indoors or out. A 4-foot cable with standard mini-stereo plug and a 6-foot expansion cable with a standard (1/4-inch) stereo phone plug are included.

MS 100 Suggested Retail Price $85.00
CAT. NO. 1SMS100 **Cornucopia Price $75 ($4.00 P&H)**

★ Cornucopia Shopper Special ★

Get either of these two premiums for only $5 when you order ANY item from this catalog!

See HCM's center bound-in order form for merchandise details and regular prices of these items.



Cornucopia Price $5
ON DISK REVUE CAT. NO. 1EODR ($1.00 P&H)
Best Of 99'er (Book Only) CAT. NO. 1EB99 ($1.50 P&H)

The Great Deception

All CDs are created equal—but some are more equal than others . . .



There's no question—the Compact Disk (CD) player is the hottest home electronics product to come down the pike—ever! The CD is the most revolutionary change in audio since the invention of the LP phonograph record. With the new optical medium, the days of scratchy needles and warped records are over! Laser pick-ups read the digitally recorded music from shiny 4-inch discs without even touching the discs themselves. Your thousandth playing of the music is virtually the same quality as the first—and the music literally explodes from your sleepy stereo system, with a crystal clarity, dynamic range, and lack of noise that—without exception—startles first-time listeners . . .

But you know all this—you've been reading about this new music revolution in the popular press, and you can hardly avoid hearing all the better radio stations across the country playing the new discs. Chances are you have friends who have been raving about their new purchase . . . or maybe you've been curious enough to stop into a local store and give the new medium an "ear test."

If you've seen a few of the CD players side-by-side, one thing is obvious: they all operate pretty much the same—with smooth, motor-driven loading trays, digital readouts, random programming for making selections in any order you choose, plus pretty much the same search and skip features. The controls are all logical, intuitive to first-time users, and don't require reading a lengthy operating manual prior to hook-up or use. In short—the perfect consumer-electronic product.

But therein lies the danger to the unwary consumer: CD product development is already into its fourth generation—characterized by subtle-yet-important player improvements which open up the doors for a major change in the audio-only CD medium itself. The most advanced of these fourth-generation players must command higher prices, yet unfortunately have to compete in the marketplace with vastly inferior machines that are practically being given away at fire-sale prices. And the public is led to believe that all CD players are alike—giving rise to *The Great Deception.*

THE CODE OF THINGS TO COME

The majority of CD players being sold today are already technically obsolete—because they lack one all-important feature: a subcode output port. This will allow a CD player to hook up to your TV (through an accessory decoder/adapter box) enabling you to display still pictures (video or computer-generated) relating to the audio material, scores, liner notes, lyrics, or other alphanumeric information. Imagine a hot rock-video disc with pictures of the stars (get ready, Madonna . . .), operas with translated librettos, classical recordings with prominent orchestral sections displayed, travelogues, children's stories, and even encoded MIDI signals that trigger synthesizers and drum machines—plus orchestrate graphic light shows synced to the music! All this will eventually enhance the music content of CDs—you surely won't want to miss it by purchasing an unequipped player . . .

THE UNIQUE TECHNIQUES OF TECHNICS

Enter the Technics SL-P3, a fourth-generation CD player of exceptional competence and value. Yes, it's one of the few that has a subcode outport port; but it's also the only player we know of with several other crucial features for the audio enthusiast or electronic musician: remote-controlled volume, A-B frame marking, auto-pause, and auto-cue.



The wireless remote control unit is so complete, it even has a volume control.

The wireless (infrared) remote control is the most complete we've seen. You can perform virtually every major front-panel function from the comfort of your couch across the room—*including volume adjustment*! Because CDs contain a much wider dynamic range than other music sources—and even differ widely from disc to disc—listeners are always having to readjust their amplifier volume control setting every time they switch sources or CDs. With the SL-P3, volume level can be compensated for remotely (through special back-panel variable line-out terminals) to match the output level with your other components—leaving your usual amplifier volume level setting alone. The unit even memorizes the last output level setting when the power is switched off! Virtually every member of our staff who has seen this remote-volume feature in operation has vowed never to buy a CD player without it . . .

Then there's the A-B frame-marking capability, allowing you to mark *any* two points on the disc—not just the beginning or end of individual tracks, or indexes within tracks—for repeat play. When combined with Technics special auto-pause and auto-cue features, it allows precise punch-in and punch-out points for digital sound sampling (see page C-7) and specialized recording needs. Used alone, the auto-pause feature puts you into pause at the end of each programmed track; auto-cue puts you into pause right at the start of the first musical note in the next cut. Both are quite useful features, particularly when dubbing CDs to tape for your car deck or personal portable.

"FULLY LOADED" UNDERSTATEMENT

The SL-P3 is Technics top-of-the-line CD player. It's fully loaded with state-of-the-art features—such niceties as direct access of all tracks and index points, a music scan function with *adjustable* time interval, a disc prism for viewing the spinning disc from the front as well as above, automatic playback from an external audio timer (see page C-13), and a pre-emphasis compensation circuit (with front-panel indicator) for specially encoded CD's. Keep in mind that this full spectrum of special features just *isn't* available on other CD players—no matter what the price!

SOLID BENEATH THE SKIN

The SL-P3 rates right among the very best in tracking and error correction—the most critical performance specs for the CD medium. It lets the SL-P3 play low-quality CDs—or CDs with defects such as air bubbles and scratches—that would cause skipping or noise on many other units. And with its optical deck mounted in a protective floating insulator, the SL-P3 has excellent shock resistance and reliability.

UNEQUALED VALUE

How much would you expect this fine CD player to cost? Technics designed it to sell for $700. Yet due to the very competitive nature of the audio market, we were able to make a special purchase, and so can now offer you this unit for *half* this figure!

Sure, you can buy cheaper CD players, but none of them hold a candle to the SL-P3. And quite frankly, we turned down a slew of less expensive units for this catalog—we just didn't feel good about selling obsolescence and inferior value . . . So, if you're ready for a revolutionary audio experience, do yourself a favor and seriously consider purchasing this CD player from us. You'll be glad you did.

SL-P3 Suggested Retail Price $700.00

Cornucopia Price
CAT. NO. 1TSLP3 **$349 ($12.00 P&H)**

CORNUCOPIA BOOKSTORE



SYNTHESIZER BASICS

Keyboard Library
— Hal Leonard, Publisher

Explore and discover the basic principles of synthesis and recording. Reprints of articles written by Robert Moog, Roger Powell, Tom Rhea, and others offer a wealth of advice and information. Step-by-step explanations and graphic illustrations help clarify technical points. An introduction to MIDI is also included.

Cornucopia Price $8.95
CAT. NO. 1HL01 ($2.50 P&H)

HOW TO MAKE AND SELL YOUR OWN RECORD

— Diane Sward Rapaport

Planning your record or tape production takes more that a good tune and recording equipment. Get the full story on production, packaging, marketing, copyrights, and more with this revised and enlarged edition. Information on expenses, time management, cassette duplication, new technologies, and foreign licensing are also included.



Cornucopia Price $12.95
CAT. NO. 1HP01 ($2.50 P&H)



SYNTHESIZER TECHNIQUE

Keyboard Library
— Hal Leonard, Publisher

This volume details re-creating timbres, pitch-bending, modulation, expression, soloing, and orchestration. Contributions by Robert Moog, Tom Coster, George Duke, and others give you practical instruction on mastering the technical and artistic potential of your synthesizer.

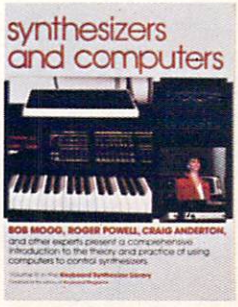
Cornucopia Price $8.95
CAT. NO. 1HL02 ($2.50 P&H)

THE COMPLETE GUIDE TO SYNTHESIZERS, SEQUENCERS & DRUM MACHINES — Dean Friedman

Your "bible" to the realm of electronic keyboard and sound equipment. This fully illustrated book shows you what's on the market, what they do, and how much they cost. It gives you valuable information and helps you make informed and confident purchasing decisions. Also included is a summary on music software, a glossary, and a list of manufacturers.



Cornucopia Price $9.95
CAT. NO. 1AMSC01 ($2.50 P&H)



SYNTHESIZERS AND COMPUTERS

Keyboard Library
— Hal Leonard, Publisher

Open your music studio to the world of computers. This book provides a comprehensive introduction to using computers to control synthesizers. Get an overview of digital synthesis, FM synthesis, Fourier synthesis, and digital sampling. A complete section on MIDI helps you build a solid understanding of this computer-assisted music revolution.

Cornucopia Price $8.95
CAT. NO. 1HL03 ($2.50 P&H)

Cornucopia



THE PLATINUM RAINBOW

— Bob Monaco, James Riordan

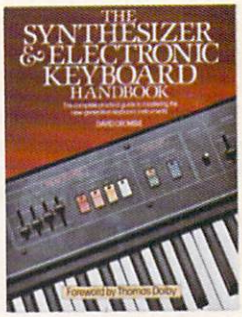
A realistic look at the music and recording industry. Learn how to find a manager, producer, and agent; promote yourself; get free recording time, and *much* more. This newly revised and updated edition includes a complete directory of everyone you would ever need to contact. It's a must for anyone pursuing a music career.

Cornucopia Price $9.95
CAT. NO. 1RPR1 ($2.50 P&H)

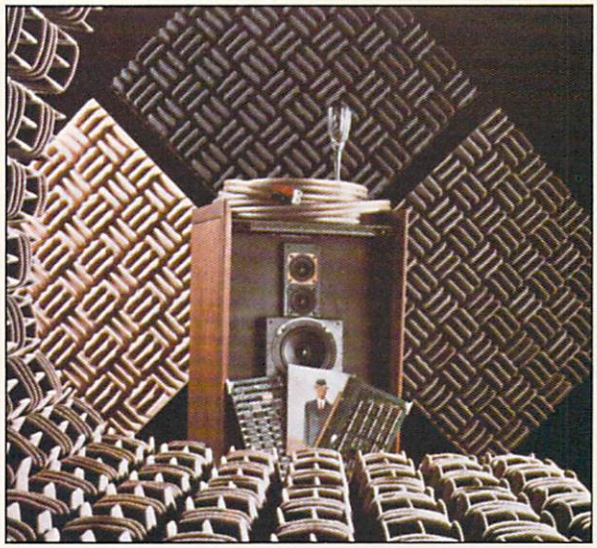
THE SYNTHESIZER & ELECTRONIC KEYBOARD HANDBOOK

— David Crombie

This is the answer to your questions about keyboard instruments. Crombie's guide covers the whole spectrum of instruments, playing techniques, music theory, MIDI, and more. Illustrations, photos, and in-depth explanations make mastering keyboards a breeze. Plus, a special section profiles the most influential and innovative keyboard players of our time!



Cornucopia Price $13.95
CAT. NO. 1RH01 ($2.50 P&H)



Listen To The Sound Of Silence!

SONEX Juniors Acoustic Foam

SONEX Juniors employ a special anechoic wedge pattern for ideal sound absorption at all audio frequencies. These attractive 24-inch square panels smooth frequency response and reduce annoying standing waves. Convenient adhesive mounting squares and complete instructions come with each box of 4 panels. Available in 4 appealing colors: natural charcoal, brown, blue, and beige.

Cornucopia Price $48 ($4.00 P&H)

Blue	CAT. NO. 1SBLU	Charcoal	CAT. NO. 1SCHAR
Brown	CAT. NO. 1SBRN	Beige	CAT. NO. 1SBGE

Cornucopia™

Distinctive Products for Music & Electronics

1500 Valley River Dr., Suite 250
P.O. Box 70288
Eugene, Oregon 97401

ORDER TOLL FREE

FOR CREDIT CARD ORDERS CALL

1-800-828-2212

MONDAY through FRIDAY 7:00AM — 4:00PM P.S.T.
SATURDAY 8:00AM — 1:00PM P.S.T.

—Please have your credit card information and catalog ready before calling. We suggest that you use this order form to organize your phone order before calling our toll-free number.

For Inquiries & Order Status:
Tel. (503)-341-1029
MONDAY through FRIDAY 8:00AM — 5:00PM P.S.T.

HOW TO ORDER: Just fill in our item number, the catalog page, the item description, and quantity. Then fill in the amount for postage and handling which you will find in parentheses (P&H) after the price of each of our products in this catalog.

***To ensure that our prices are kept as low as possible, and to provide the speediest service available, we cannot accept personal checks or C.O.D. orders as payment for merchandise. If not charging to a credit card, please obtain an appropriate Money Order or Cashier's Check made out to "Cornucopia." Thank you for your understanding and cooperation.**

IMPORTANT INFORMATION

1) **Shipping**—Whenever possible we will ship your order via UPS. **Orders shipped to Continental U.S. only.** Please give us a street address rather than a P.O. Box. Most orders shipped within 5 working days. In the unlikely event that your order cannot be fulfilled within 30 days, we will notify you and give you the option to cancel. Items ordered together are not necessarily shipped the same day. Your credit card will be billed only for items that have been shipped. For over-size items, we ship air express to prevent unnecessary delays which would otherwise be imposed by truck shipment. These items are designated by catalog numbers ending in "F."

2) **Manufacturer's Warrantees**—All products sold by Cornucopia are covered by the individual manufacturer's warrantees. If you would like to see a copy of an exact warranty before you buy, just send us a letter (do not phone) with a postage-paid return envelope giving us the product number and name to the attention of the Warranty Department.

3) **Misprints**—Cornucopia makes every effort to ensure that the information in this catalog is correct. We cannot be responsible for errors. However, should you buy a product based on our error, it is of course covered by the normal conditions of our 30-day free-trial period.

4) **Prior Sale and Prices**—Cornucopia deals with dozens of manufacturers. Sometimes we simply sell more than a manufacturer can make. In any event, we cannot guarantee to supply any product advertised. We will immediately refund your money if we find that we can no longer obtain a product or if its price rises.

5) **Purchase Orders**—We accept written purchase orders from schools, institutions, and the government. Invoices must be paid in full prior to shipment.

THANK YOU FOR YOUR ORDER

If you prefer to leave the catalog intact for reference or future orders, you may photocopy this order form.

Order Form

☐ **Cashier's Check or Money Order enclosed***

Charge my ☐ **VISA** ☐ **MasterCard** **Date Expires** ____________

Account No. ____________

Signature ____________ **Tel. No.** ____________

PURCHASER

Name ____________

Address ____________

____
TO ENSURE FASTER DELIVERY, STREET ADDRESS IS ESSENTIAL

City ____________ State ________ Zip ________

ALTERNATE SHIPPING ADDRESS

Please give name, address, and item no. for each item to be shipped to an address other than your own.

Item No. ____________ ____________ ____________
____________ ____________ ____________

Name ____________

Address ____________

____
TO ENSURE FASTER DELIVERY, STREET ADDRESS IS ESSENTIAL

City ____________ State ________ Zip ________

ITEM NO.	CAT. PAGE NO.	ITEM DESCRIPTION	ITEM PRICE	QTY.	DELIVERY (P&H)	TOTAL CHARGE

TOTAL $

FOR OFFICE USE ONLY

ORDER INITIAL ____________

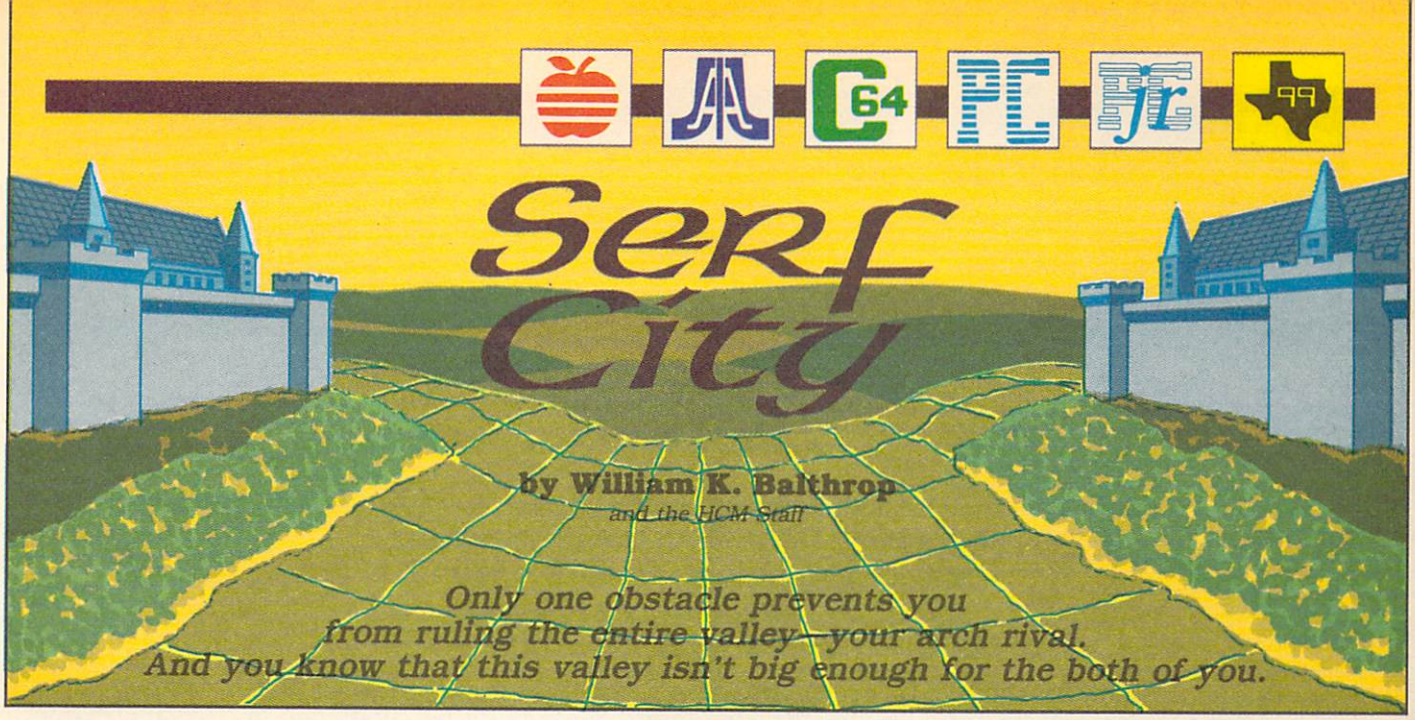
METHOD OF PAYMENT

☐ BANKCARD ☐ CASHIERS CHECK/MONEY ORDER

UPS No. ________ INS ________ DATE __/__/__ $ ________ INITIAL ________

________ ________ __/__/__ ________ ________

________ ________ __/__/__ ________ ________



The year is 985 A.D. From a balcony just outside your royal quarters, you survey your entire kingdom in the valley below. Across the valley sits the castle of your arch rival. For years, you and the other king have shared this valley, battling for land and food for your people. But now you tire of this status quo—you want the valley all to yourself. Your mind searches for a means to this end. An all out battle between armies? A fight-to-the-death between you and the other king? No. Experience has taught you that only through shrewd economics can you slowly weaken your opponent and rule the entire valley.

Serf City is a two-player economics simulation employing both text and graphics. Each player rules one of the valley's two kingdoms, both largely subsistent on wheat crops. Your objective is to increase the size, wealth, and population of your kingdom until you run your opponent out of the valley.

But this is no shoot-em-up arcade game. Only sly economic tactics can put you on the road to prosperity in this simulation. Several devices are at your disposal: You can purchase land, establish commerce, build wheat mills, plant crops, sell wheat, raise taxes, and hire or fire your own army. However, you don't have control over everything! The program contains random factors simulating unforeseen disasters that can severely diminish your kingdom's crops or population. (See Figure 1 for a schematic of how all these economic factors interrelate.)

Factors Of The Realm

When the program starts, you and your opponent must enter your names. Then, the main data display appears on the screen (see Photo 1). The data screen displays the current year—beginning in the year 985 A.D.—next to your name. A turn lasts one year, but the number of years that a game can last is unlimited. A game terminates when the players quit or when one player has no more population or land.

The main data display also lists your initial economic resources. Both players begin the game with 12 units of land, 90 people, 500 gold pieces, and 500 bushels of surplus wheat. The land is occupied by 2 units of commerce and 1 mill, leaving 9 units for farming.

After you've briefly studied the data display, you begin play by pressing the key corresponding to the economic device you want to manipulate. It's time to don your royal economic thinking cap and commence your most cunning ruling strategy! Prompts guide you through the rest of the simulation. When you complete entering data for the current year, press [RETURN] or [ENTER]. Now, it's your opponent's turn. When your opponent has finished entering data, press [RETURN] or [ENTER] again to display the year-end economic report, which is based on the decisions made during your last turn (see Photo 2). The program then shows your updated score on the main data display during your next turn.

Population

Population varies each year, due to births, deaths, immigration, and emigration. And these factors fluctuate depending on your kingdom's overall prosperity—a healthy food supply, reasonable tax rates, and a sufficient work force. To increase your population, for example, feed your people more (see "Food" section), making sure that you have enough field and mill workers to produce the extra food you need. Feeding your people less reduces your population. Population can also drop when the personal tax rate is too high (see "Personal Tax" section); people may move to other kingdoms with lower taxes (but never into your opponent's kingdom). If your population drops to zero, your kingdom falls, and your opponent rules the entire valley.

Employment

The program allocates portions of your population to work in your kingdom's fields, mills, and commerce. Field workers are your highest priority—without them you have no wheat to process for food. Therefore, the program automatically allots enough people to harvest all of the land you have planted. If you have sufficient population, the program then allocates any additional people to work in mills. Next, the program makes commerce workers out of any other people you have after fully staffing your fields and mills. Finally, the program considers any leftover population to be unemployed.

Commerce

The program expresses all merchant trade in terms of *units* of commerce, each unit costing 500 gold pieces and occupying one unit of land. Your compensation is the benefits of taxation (see "Commerce Tax" section).

You can place only one unit of commerce on each unit of land not occupied by mills. If your population is too small for the number of commerce units you own, some



Photo 1
From the main data display, you can select and manipulate any of the economic devices you have at your disposal.

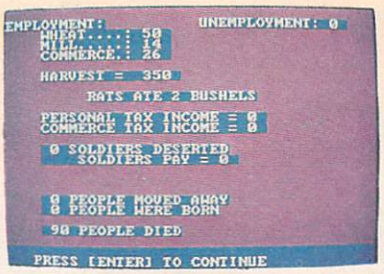


Photo 2
After each turn, the program displays the year-end economic report, listing the status of the economies of both kingdoms.

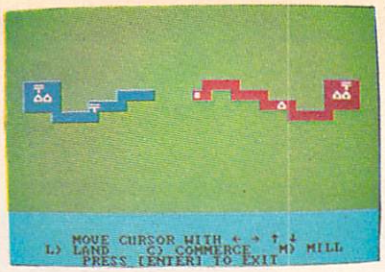


Photo 3
The map of the valley depicts land, mill, and commerce ownership by painting each kingdom a different color.

shops must close, thus reducing the flow of taxes.

To buy units of commerce, press [M] to select the Map option from the main data display (see Photo 1). A map of the valley appears, depicting your land and your opponent's land in different colors (see Photo 3). Position the cursor on top of the land you already own (see appropriate Control Capsule for cursor movement keys). Press [C] for Commerce, and a graphic symbol appears in place of the cursor to indicate the new addition. But, if you don't have the 500 gold pieces required to establish a unit of commerce, the symbol does not appear.

"It's time to don your royal economic thinking cap and commence your most cunning ruling strategy!"

Mills

Mills produce flour from the wheat you buy or harvest. A larger population requires more mills. Without at least one operating mill, all your people will starve. On the other hand, running too many mills pulls people out of commerce, reducing commerce-tax revenue.

A fully staffed mill (14 workers) produces enough food for 100 people. Understaffed mills produce less wheat—or with no workers, shut down entirely.

A new mill costs 1,000 gold pieces. To construct a mill, press [M] from the main data display to view the Map. Move the cursor on top of your own land, press [M] for Mill, and a graphic symbol representing the new mill appears on you land. Build as many mills as you want (only one per land unit), as long as you have enough open land and lots of gold!

Food

Your food supply relies on 3 factors: the number of mills operating; the number of people available to work in the mills; and the amount of wheat available for consumption.

Without enough food, your people will starve or move to another kingdom, reducing your personal-tax revenues (see "Personal Tax" section). Feeding your people slightly more than they need results in more births, fewer deaths, and an influx of people to your kingdom—which increases personal-tax revenue. However, large-scale over-feeding results in too many births, too few deaths, and too many emigrants—which eventually leads to over-population and an unstable economy. Overpopulation leads to high unemployment, with people depleting the food supply but not working to compensate for what they consume.

To decree the amount of wheat available for food, press [W] to select the Wheat option from the main data display. When the Wheat option menu appears, select Feed the People, and enter the number of bushels you want to feed your kingdom. Maintaining your current population requires a supply of at least 5 bushels of wheat per person.

Wheat Surplus

Surplus wheat includes all the wheat you harvest or purchase each year. As in real life, your yearly harvests vary. *Serf City* contains a random factor simulating yearly variances in harvests, such as bumper crops due to favorable conditions, or crop failures due to swarms of locusts.

You can allocate your surplus wheat however you wish—sell it for gold, plant it as seed, process it as food, or put it in storage. You may even want to use wheat as a commodity, buying bushels when the wheat price is low and selling when the price is high. But if you leave your wheat in surplus too long, rats might eat large portions (possibly all) of it.

Wheat Storage

Store your wheat that is not being used. To store wheat, press [W] to select the Wheat option from the main data display. When the Wheat option menu appears, select Wheat in Storage, and enter the *total* number of bushels you want to store. For example, if you want to add 20 bushels of wheat to the 50 bushels you already have in storage, simply enter "70." To take all of your wheat out of storage, enter "0" for the amount of bushels in storage, and the program automatically moves all your wheat into surplus. Now you can allocate the wheat any way you want.

Seed Planted

The amount of wheat you plant as seed determines the size of next year's harvest and the number of people required to work in the fields. You can plant up to 10 bushels on each unit of plantable land—land not occupied by mills or commerce. Planting more than 10 bushels per plot of land wastes the excess wheat seed.

Each bushel of seed planted requires one person working in the fields. Therefore, planting too much seed pulls workers from mills and commerce to work in the fields. And with no mill workers, your kingdom starves—even if you have a bumper crop! Let's say you have a population of 100 and plant enough wheat to put 95 workers in the fields, leaving only 5 people to work in the mills. These 5 workers cannot produce enough food to adequately feed your entire kingdom.

To plant seed, select the Plant Seed option from the Wheat option menu, and enter the desired amount.

Total Land

This total on the main data display represents all of the land you own. The only restrictions on the amount of land you can own are the dimensions of the screen and your own financial resources. The cost of land varies from 25 to 150 gold pieces—higher wheat prices raise the price of land.

Remember that every time you add a unit of commerce or a mill, you have one less unit of plantable land. Therefore, while expanding your kingdom, be sure to purchase enough land for crops to feed your people.

To acquire land, move the cursor to a plot of land that is *adjacent* to land you already own, and press [L] for Land. If you have enough gold to pay for this land, your color appears at the cursor's location.

Soldiers

You need soldiers to capture land from your opponent and to protect your own land from similar attacks. To seize a unit of your opponent's land, you must expand your kingdom's boundary until it touches your rival's boundary. Then move the cursor on top of your opponent's land (adjacent to your land), and press [L]. If you have enough gold to pay for the land, and enough well-paid soldiers to defeat the opposing army, the land is yours. However, as in any war, you pay a price—you lose a random percentage of soldiers.

To hire or fire soldiers, press [S] for Soldiers on the main data display, select either Hire or Fire, and enter the number you desire. Keep in mind that hiring soldiers pulls people out of your work force. Creating a large army to protect your lands may not leave enough people to work in the fields, mills, or commerce. On the other hand, too few soldiers may be an open invitation for your opponent to seize *your* lands.

Soldiers' Pay

You must pay every soldier an annual fee. The more you pay your soldiers, the harder they fight in battle. However, paying one soldier 100 gold pieces is not as wise as paying 5 soldiers 20 gold pieces each. And even in the 10th century, soldiers have their standards—pay your soldiers too little, and they will desert your kingdom, thereby reducing your population.

To set army wages, select the Pay option from the Soldiers option menu, and enter the amount you want to pay each soldier per year.

Gold Pieces

Gold is essential to the financial success of your kingdom. You can use gold to purchase land, wheat, mills, and commerce. You can acquire gold through taxation (see "Personal Tax" and "Commerce Tax" sections), or by selling wheat you have already harvested (see Wheat Surplus section).

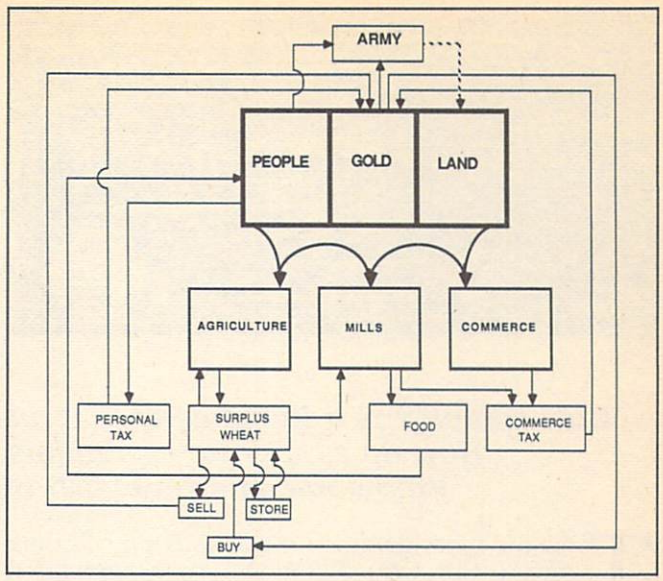


Figure 1
This diagram illustrates some of the intricate interrelationships that you must control to preserve and enrich your serf-based economy.

Commerce Tax

This tax is placed on your mills and the merchants engaged in commerce. Commerce-tax revenue varies, depending on the number of workers in mills and commerce and the prosperity of your people.

The tax also affects the number of people that can work in commerce and mills. For example, if you set the commerce tax at 10 percent, the merchants can employ 22 people per unit of commerce. At 30 percent, your merchants can afford only 18 employees per unit of commerce. Therefore, if you set the tax rate too high, the unemployment rate rises as some shops must close. Set the rate too low, and your revenues diminish. The maximum commerce-tax rate is 69 percent—commerce will not tolerate a higher tax rate.

Personal Tax

You can also establish, raise, and lower a personal tax to generate revenue based on population and the people's prosperity.

When over-taxed, your people spend less, reducing your commerce-tax revenue. Over-taxing may also drive people away and cause more deaths and fewer births.

To set the personal tax rate, select [T] for Tax from the main data display, press [1] to select the personal tax rate, and enter the percentage you desire. Like the commerce tax, the maximum personal tax rate is 69 percent—any higher and the people will revolt.

For your type-in listings, see HCM PROGRAM LISTINGS CONTENTS. **HCM**

CONTROL CAPSULE

Serf City

KEY	FUNCTION
E	Move map cursor up
S	Move map cursor left
D	Move map cursor right
X	Move map cursor down
ESC	Save/exit game
RETURN	Exit current option

CONTROL CAPSULE

Serf City

KEY	FUNCTION
↑	Move map cursor up
←	Move map cursor left
→	Move map cursor right
↓	Move map cursor down
ESC	Save/exit game
RETURN	Exit current option

CONTROL CAPSULE

Serf City

KEY	FUNCTION
E	Move map cursor up
S	Move map cursor left
D	Move map cursor right
X	Move map cursor down
F1	Save game
F7	Exit game
RETURN	Exit current option

CONTROL CAPSULE

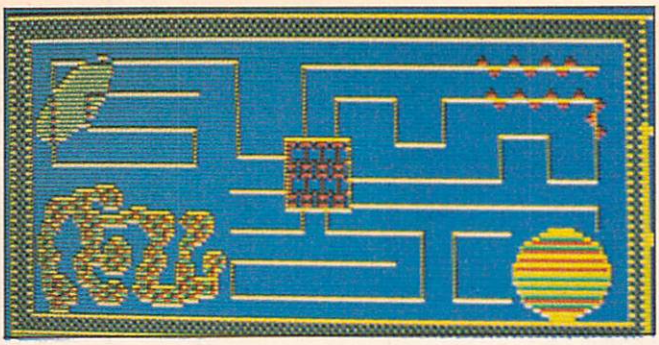
Serf City

KEY	FUNCTION
↑	Move map cursor up
←	Move map cursor left
→	Move map cursor right
↓	Move map cursor down
ESC	Save/exit game
ENTER	Exit current option

CONTROL CAPSULE

Serf City

KEY	FUNCTION
E	Move map cursor up
S	Move map cursor left
D	Move map cursor right
X	Move map cursor down
FCTN 9	Exit game
ENTER	Exit current option



CELL MATES

by William K. Balthrop
& Wayne Koberstein
HCM Staff

How complex is a living cell? Is it a static structure—or a dynamic system? How do symbiotic organisms make all cellular life possible? Find some answers with this fascinating bio-simulation.

There is more going on inside a living cell than meets the eye. Even the most powerful microscopes have not yet revealed the entire story—the secret life of the cell. Ask most people what they know about this basic life unit, and their reply, if any, may invoke a static picture of the cell and its interior: the cell wall, the nucleus, and "a bunch of little floaty things." But when biologists look closer at these little floaty things, they see participants in an incredibly complex, wonderfully cooperative, and dynamic *ecosystem*; for the metabolism of a living cell involves some strange and alien participants who apparently learned to live together millions of years ago. Directing this metabolic dance, communicating with each cell part through encoded molecules, is the nucleus. Some of the participants—called *organelles*—possess their own DNA (deoxyribonucleic acid) and reproduce themselves independently from the nucleus. Together, they are all *Cell Mates*.

What if you could place yourself at the center of this dance—consciously balancing the entire system moment by moment? There may be no better way to appreciate just how dynamic and basically cooperative this system is. We can't literally perform such a miracle, but we *can* give you *Cell Mates*, a simplified computer simulation that places you in the role of the nucleus, the cell's "choreographer."

The Major Performers And You

Plant and animal cells each have a somewhat different cast of dancers. But in both cases, each kind of organelle has a specific role to perform. In animal cells, some organelles, called *ribosomes*, manufacture various proteins. Another group, the *Golgi apparatus*, "packages" these proteins for internal use or secretion (export). *Lysosomes* are the garbage collectors. And tiny mobile chemical plants, called *mitochondria*, supply energy for all of these activities.

Although these organelles perform the cell's main functions, there are many other participants performing minor, but essential parts. Because of the difficulty in portraying all of the cell's diverse components, these minor players are beyond the scope of *Cell Mates*.

In *Cell Mates*, you direct the show, allocating energy to these 4 essential functions: protein production, waste disposal, ribosome production, and protein export. Your goal is to direct these functions in such a manner that the cell is able to divide—effectively producing two new cells in its place. When the cell divides, you receive a total score. The less time you take to reach cell division, the higher is your score.

You must meet certain requirements before the cell can divide: You must double the supply of protein as well as the number of several internal components (such as the ribosomes); and you must support the larger family of surrounding cells by exporting protein.

After displaying the title, the program presents the actual playing screen. This screen contains a graphics representation of the cell (in the upper-left corner), and 12 indicators that show the status of the various processes in the cell. You can gauge your progress by viewing these indicators. Some of the indicators contain areas highlighted against the background color. To maintain and eventually reproduce the cell, you must manipulate the cell's life processes so that the pointers remain in these "safe" areas. Indicators that contain no distinguished area show the actual level of energy going to specific activities.

Special thanks for technical advice goes to W.R. Sistrom, University of Oregon Department of Biology.

THE SECRET LIFE OF ORGANELLES

Millions of years ago, life wasn't what it is today. In the still waters of land-locked lakes and ponds, primitive single-celled creatures without true nuclei learned to obtain energy and reorganize materials through ingenious chemical processes. These were the ancestors of the organelles; for as life became more and more organized, such specialized organisms became permanent fixtures within higher lifeforms. Some—such as the *chloroplasts*—integrated themselves into the first true plants, capturing energy from the sun for use by the entire organism. Others—such as the *mitochondria*—took oxygen from the atmosphere (mostly produced by the activity of the chloroplasts) and used it to "burn" complex molecules as an energy source for animal life. Even today, these organelles possess their own DNA, and reproduce themselves independently of the organisms they support. (For example, mitochondria in a fertilized egg cell come partly from the mother cell and partly from the sperm—and as this cell divides, each new cell starts off with half the mitochondria of a mature cell.)

Why Life chose to organize itself in this way is a question that has preoccupied many people—not just the biologists who discovered these facts. For whatever reasons, the organizing process was primarily *cooperative*, rather than competitive. It is still an open question whether organelles live for the sake of the higher organisms—or whether we are living for them. What is apparent is that all the players in this game have a total stake in the system. Neither the organelles nor the organisms in which they reside could live without each other. The fact that each maintains a separate genetic indentity actually serves to dramatize the cooperative nature of their relationship. Of all life's mysteries, this may be one of the most curious and wonderful.

[For more on this subject, read *The Lives of A Cell* by Lewis Thomas.—Ed.]

We have incorporated a delay factor into the simulation, representing the real-world lag between the initiation of an action and its effect. This delay is not constant—it varies with internal and external conditions. Because of this built-in lag, any action that you take does not immediately show up on the screen indicators.

> *". . . when biologists look closer at these little floaty things, they see participants in an incredibly complex, wonderfully cooperative, and dynamic ecosystem."*

Dividing Energy

This simulation focuses on the 5 most important cell components: the mitochondria, the nucleus, the lysosome, the ribosome, and the Golgi apparatus.

Mitochondria—In animal cells, mitochondria produce ATP (adenosine triphosphate)—life's most available source of chemical energy. Similar organelles in plant cells—the *chloroplasts*—actually perform photosynthesis (the conversion of light into energy). All of the cell components, including the mitochondria themselves, use ATP as their energy source. Without this substance, the cell would not be able to carry out any activity. Mitochondria manufacture ATP by combining products of available nutrients with oxygen (and emitting carbon dioxide as a waste product). Thus, when cellular biologists talk about *respiration*, they are talking about a process that occurs inside the mitochondria. The amount and quality of available nutrients directly affect the amount of ATP that the mitochondria can produce.

Mitochondria are mobile, and can move to the areas of the cell where energy is most needed. Most animal cells contain anywhere from one to hundreds of thousands of mitochondria. For the sake of simplicity, our cell representation contains only one. To increase the energy allocation to any of the cell's components, simply move the mitochondrion to the corresponding organelle symbol by pressing the appropriate key (see the Control Capsule). Once the mitochondrion reaches the designated organelle, it automatically starts producing ATP. The mitochondrion does not manufacture ATP while it is in transit between cell parts.

Nucleus—The nucleus is the cell's control center—the "central computer." It sends signals in the form of encoded molecules to each part of the cell, telling it to carry out various operations. (In addition, the nucleus carries genetic information in its DNA that determines the form of new cells resulting from cell division, or "mitosis.") Again for simplicity, we have symbolized the message-sending activity of the nucleus with one example: the message to increase production of ribosomes. Of course, the nucleus requires energy each time it sends a message.

To allocate more energy to the production of ribosomes, press N. The mitochondrion responds by migrating to the nucleus. Once the mitochondrion reaches the nucleus, it begins producing ATP. The increase in energy shows up on the screen indicator labeled RIB.ENERGY (energy for the ribosome-production message).

As you allocate this energy, the RIBOSOMES pointer shows an increase by moving to the right. If you direct the mitochondrion away from the nucleus, the RIB.ENERGY indicator shows a decline, and the RIBOSOMES indicator ceases to increase.

Ribosomes do not live forever. They die occasionally, and become part of the cell's waste. You need to replenish your supply of ribosomes occasionally to maintain their numbers.

Ribosomes—Ribosomes produce protein. These tiny organelles are attached to the *endoplasmic reticulum*—a convoluted membrane extending out from the nucleus. The ribosomes assemble various proteins according to patterns supplied by the nucleus. Although we symbolize the ribosomes in a discrete corner of our cell, in reality they are more evenly distributed.

Protein is the cell's prime material. Before a cell can reproduce, it must double its original supply of protein. The speed at which ribosomes produce protein depends on the total ribosome quantity. If you need to produce protein at a faster rate (to keep up with the protein exportation rate, for example), you must increase this quantity. (See the Nucleus section above for directions on increasing the ribosome count.)

Press R for Ribosomes to allocate energy to the production of protein. This signals the mitochondrion to migrate toward the ribosomes (upper-right corner of the cell area). Once the mitochondrion reaches the ribosomes, the PRO.ENERGY indicator shows an increase as the protein energy level ascends. This increase in

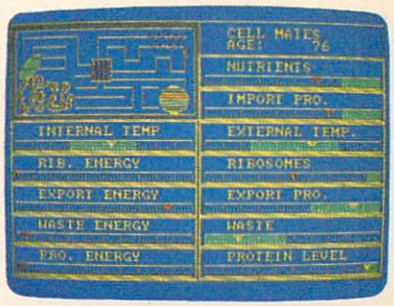


Photo 1
Cell display from IBM version at top left shows (clockwise from display's upper-right): protein-producing ribosomes; waste-eliminating lysosome; and the convoluted Golgi apparatus where the mitochondrion (green) is supplying ATP energy for protein export. Nucleus is at center with extending endoplasmic reticulum membrane.

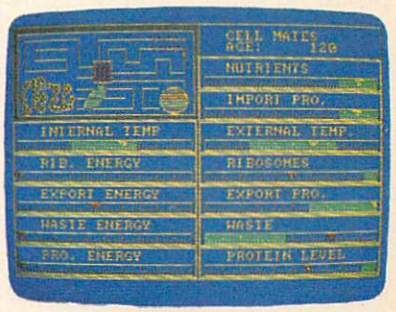


Photo 2
As the waste level continues to increase (as shown by waste indicator), the mitochondrion produces no ATP while traveling from the Golgi apparatus to the waste-eliminating lysosome at lower-right.

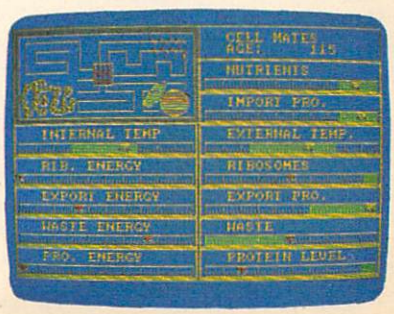


Photo 3
Now at the lysosome, the mitochondrion's ATP energy goes toward eliminating cell waste products—and in the nick of time, waste levels begin to decrease.

energy shows up in the PROTEIN indicator. But, as we stated, to achieve the desired increase in protein, you may need to increase the number of ribosomes first.

Lysosomes—The lysosomes are the cell's housekeepers. They carry degradative enzymes and travel around inside the cell collecting and digesting larger chunks of waste composed mostly of worn-out cell parts. This waste is always accumulating, so you must frequently budget energy for lysosome activity.

As the amount of waste in the cell increases, the WASTE pointer moves to the right. The only way to reduce the level of waste is to increase the amount of energy allocated to waste disposal. To do this, press L for Lysosome. In response, the mitochondrion moves to the lysosome symbol at lower-right corner of the cell. Once the mitochondrion arrives, the energy level shown by the WASTE ENERGY indicator increases. After the energy reaches a sufficient level, the amount of waste starts to decrease as the lysosomes do their job.

Golgi Apparatus—Your cell is also in the import/export business. Few cells can survive without the aid of other cells, so you must not only support your own cell, but those cells surrounding you. You must supply these cells with the protein they need for their existence, and in return you will receive protein from them—some of the nutrients vital to the production of ATP. The Golgi apparatus is responsible for exporting protein to other cells. As a secondary role, the Golgi apparatus spends a small percentage of its total energy packaging up waste and sending it out of the cell. If you fail to export sufficient protein, both the level of imported protein (IMPORT PRO.) and the nutrient level (NUTRIENTS) decline.

"Tiny mobile chemical plants, called mitochondria, supply energy for all of these activities."

To increase the amount of protein that you export, press G for Golgi apparatus. The mitochondrion responds by moving towards the lower-left corner of the cell area to the Golgi apparatus symbol. Once the mitochondrion reaches the Golgi apparatus, EXPORT ENERGY begins to show an increase. If you have a supply of protein, EXPORT PRO. also shows an increase.

The rate at which you can export depends entirely on how much protein is available. As you export protein, PROTEIN LEVEL declines. As the protein level decreases, EXPORT PRO. starts to decrease. All of the protein you export must come from the protein you have produced (shown on the PROTEIN LEVEL indicator).

Additional Factors—The AGE display indicates the passage of a relative amount of time. The AGE counter does not represent any real time frame—to make the simulation playable, we have accelerated certain aspects of the cell's metabolic system, while slowing down others.

The older the cell gets, the harder it must work to produce protein (or ribosomes). You may also notice that the average amount of nutrients and imported protein also drops off with age. Many factors affect an individual cell's lifespan—too many to explain here. But every living cell has a built-in "time limit": The cell must reproduce within a specific period of time, or it will die. Our model reflects this basic fact.

The NUTRIENTS indicator shows the cell's current food supply. The mitochondria use these nutrients to manufacture ATP. In order for the cell to reproduce, the pointer must be in the highlighted portion of the display.

The IMPORT PROTEIN indicator shows how much protein is available from the surrounding cells. This protein contributes to the cell's protein level. In order to reproduce the cell, you must move the pointer to the the highlighted area.

The INTERNAL TEMP indicator shows the cell's current temperature. The optimum temperature is directly at the center of the indicator. The highlighted area is the safe zone for the cell. If the pointer leaves this area, the cell dies. Even if the pointer remains in the safe zone, high and low temperatures (relative to center) affect the efficiency of every metabolic operation.

Most individual cells have little control over internal temperature when the external temperature goes outside a life-supporting range. Therefore, the "generic" animal cell that we depict in this simulation depends on the temperature of the tissue of which it is a part. If the cell doesn't adequately perform its part in this tissue structure—neither maintaining itself, nor supplying needed protein to the surrounding cells—its external environment tends to degrade. This can cause a drop in external temperature, as well as a shrinking nutrient supply. Normal cell activity usually maintains a safe internal temperature. If the energy levels for ribosomes, export, waste, or protein increase, the internal temperature also increases slightly. But it is more important to ensure a steady external temperature by exporting enough protein to maintain the health of the entire tissue. If the external temperature does become excessive due to high mitochondrion activity, you can press M to send the Mitochondrion to its neutral, inactive corner.

Of course, the external temperature sometimes goes outside a safe range, no matter what an individual cell does. If it remains outside this range, the cell will die—which, as they say, is Life.

Life Goes On

A BASIC computer program can only "scratch the surface" of a living cell's complex nature. While being purely fun to operate, *Cell Mates* should also serve to awaken your curiosity about this fascinating subject. Perhaps just knowing that your very life depends on the activity of microscopic creatures with their own separate genetic identities will contribute to your curiosity. As biologists continue to unravel the mysteries of cells in general, and organelles in particular, you may be in a better position to understand and appreciate this new knowledge. Here's to a lifetime of learning about life itself!

CONTROL CAPSULE
Cell Mates

Key	Function
M	Move mitochondrion to home position—upper-left corner of cell area
G	Move mitochondrion to Golgi apparatus—lower-left corner of cell area
L	Move mitochondrion to lysosome—lower-right corner of cell area
R	Move mitochondria to ribosomes—upper-right corner of cell area
N	Move mitochondria to nucleus—center of cell area
Q	Quit option—exit simulation

HCM Glossary terms: ATP, DNA, ecosystem, endoplasmic reticulum, enzyme, Golgi apparatus, lysosome, organelle, metabolism, mitochondrion, nucleus, protein, respiration, ribosome, symbiotic.

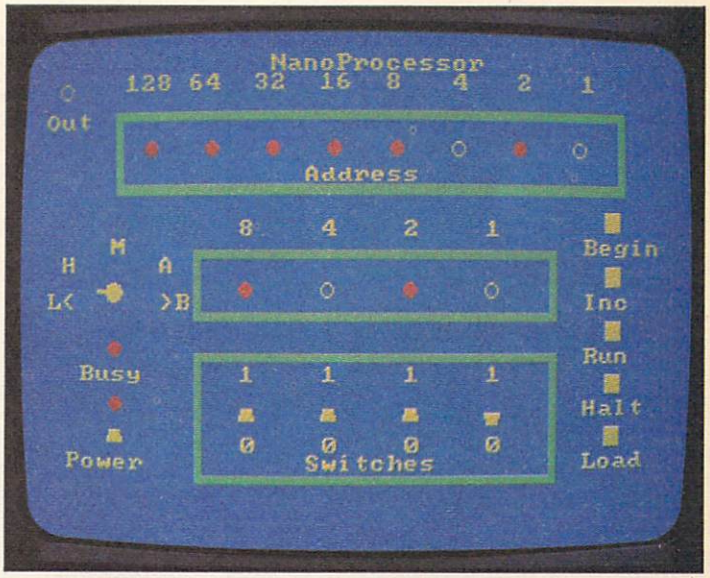
For your type-in listings, see HCM PROGRAM LISTINGS CONTENTS.

HCM

The NanoAssembler

by Roger Wood
HCM Staff

This companion to NanoProcessor shows you how an assembler can provide easy access to machine language—by translating simple instructions into the computer's native tongue.



In the last issue (*HCM* Vol. 5, No. 5), we presented *NanoProcessor*, a program that introduced the concepts of machine-language programming. This program demonstrated how a microprocessor works at its most fundamental level. Although entering and running simple programs on the *NanoProcessor* can be fun, longer and more complicated machine-language routines are another story. Even with short programs, you probably discovered what a time-consuming and error-prone process it can be to enter machine language one bit at a time.

To alleviate the difficulties involved in working with machine language, early computer users created programs called "assemblers." An assembler is a human-to-machine translator. It operates from a "dictionary" of mnemonics (a combination of letters that humans can understand), translating these mnemonics into the numbers of machine code. Using assemblers, you can write a program with the more easily remembered mnemonics, and let the computer create the actual machine language (the ones and zeros).

Thus, we present the *NanoAssembler*; a program that will teach you how to use assemblers. With the *NanoAssembler*, you will be able to write long, complicated programs for the *NanoProcessor* much more easily than you would using machine language.

Source Code To Object Code

You may find that many people refer to "assembly-language programs" and "machine-language programs" interchangeably, as though they were the same thing. Actually, an assembly-language program is a text file—known as a "source file"—that the computer cannot execute directly. It is simply a series of text lines comprising mnemonics, numbers, and labels. Before the computer can run such a program, the source file must be "assembled" or translated into a machine-language file—also known as an "object file."

Take a look at Sample Program 1, which you can load and run on the *NanoProcessor*. You may recognize this program, as it is identical to Sample Program 1 in the last issue. The two left-most columns, entitled Addr and Code, contain the machine language, (object code), which makes up the program. You can enter this object code bit by bit, or you can enter the more easily read and (with some training) understood assembly language (source code), contained in the Line, Label, Mnemonic, and Remark columns. The Remark column is like a REM statement in BASIC. It makes the program much easier to read and understand.

Our *NanoAssembler* package consists of two BASIC programs: the *NanoEditor* and the *NanoAssembler*. The *NanoEditor* is a simple text editor that lets you enter your program as source code and save it to disk (or alternatively tape on the Atari, Commodore, and TI computers). *NanoAssembler* can then read and translate that file into a corresponding file of object code, which you can save to disk or tape. You can then load the object code into the *NanoProcessor* and run it.

Creating A Program

We will use Sample Program 1 to demonstrate how the *NanoEditor* and the *NanoAssembler* work. To start, Load and RUN The *NanoEditor*. You begin with this menu:

```
1) EDIT
2) FILES
3) PRINT
4) EXIT
```

Choose the Edit option, which allows you to create and modify files. The *Editor* now displays the command prompt: CMD. You may enter one of 5 single-letter commands:

Command	Function
A	Add a line of text
E	Edit a line of text
D	Delete a line of text
I	Insert a line of text
L	List

To begin creating a new file—in this case Sample Program 1—press A. In response, the *Editor* displays line 001, with a flashing cursor waiting for your input. For each line of source code, the *Editor* provides a line number ranging from 001 to 200. When you enter the Add-a-line mode, the program always displays the cursor on a new line of source code—one line past the last line in memory. You can automatically advance to the next line by pressing [ENTER] or [RETURN]. To exit the Add-a-line mode, press the [ESCAPE] key (see your computer's Control Capsule if your machine does not have an Escape key).

Now enter the contents of the Label, Mnemonic, and Remark columns. Because our *Editor* is in BASIC, your text input will be slower than with a full-blown word processor. The Label column is empty in line 001 of our sample program, so press the proper key or key combination (see your Control Capsule) to tab into the Mnemonic field. (We will explain labels below.) Now type in the first instruction: LDA# 3. You must enter the text *exactly* as it appears in the listing, or the *NanoAssembler* program will not interpret the code properly. Make sure there is no space between the A and the #. You must, however, place a space between the # and the 3.

This spacing is critical because the Mnemonic field actually consists of two sub-fields; and the space acts as a separator for these sub-fields. The left sub-field is the "op-code," or instruction field, which defines the actual instruction. In line 001, the op-code is LDA#. The right field contains the "operand." The operand is either a two-nibble address or a single-nibble quantity to be loaded or stored in a register or memory location. It defines the number that the op-code is to operate on. In line 001, the number 3 (%0011) is the operand.

After you have entered the first instruction, you may tab into the Remark field. On a program as short as this one, however, you may choose to save time by omitting the remarks. Continue entering lines 002, 003, and 004 in a similar fashion.

Once you've entered part or all of the program into memory using the Add command, you can use the other editing commands. Each of these commands prompts you for a particular line number. E lets you Edit an already-existing line in memory. D allows you to Delete a line, and I lets you Insert a line. The L command lets you List up to 10 lines of a program to inspect what is in memory. If the program extends more than 10 lines beyond the beginning line number that you specify, you have the option to either continue listing more lines or quit and return to the command line.

Labels As Labor Savers

In line 005 (HERE JMP HERE), you encounter an important assembly-language tool—the "label." In the *NanoAssembler*, we define a label as a group of up to 6 alpha-numeric characters, beginning with a letter—in our example, the word HERE. Assembler programs use labels in place of numeric quantities. In this case, HERE represents the address to be JuMPed to. One major advantage of labels is that you do not have to know the actual numeric addresses used in a program. Instead, the assembler uses the labels to assign the correct address to a particular instruction for you.

Before continuing, let's clear up an area that sometimes confuses a beginner at assembly language: the difference between line numbers of a source file and addresses of an object file. Each line in a source file contains only one op-code. But when you assemble the source file into object code, the op-code may require as many as three addresses (see Figure 1 for the number of nibbles each instruction requires). Thus, a source file's line numbers and the actual addresses of the object code almost always differ. When the *Assembler* prints out its listing, the addresses and codes are located on the line just below the source code, representing the order of events during assembly.

By inspecting the two left-hand columns of Sample Program 1, you can see that the address to be JuMPed to is 6. You know this only because we have already assembled (or translated) the source code on the right into the object code on the left. If we hadn't provided the machine code, however, you would have to assemble all of the instructions to discover what address you wanted to JuMP to. The use of labels saves you from this tedious task and is one of the primary advantages of assemblers.

When you finish entering line 005 and press [RETURN] or [ENTER], a prompt tells you to enter line 6. This program has no line 006, so press the [ESCAPE] key for your machine (see your Control Capsule), and the program returns you to the command line. Now you can use the List command to see if you have entered everything correctly. If you find any errors, you can Edit the line or lines that they occur in. If you change a line, then decide that you don't want those changes, you can press the [ESCAPE] key instead of [RETURN] or [ENTER] to revert back to the original version of the line. This option is also available if you select Insert, but change your mind before finally entering the line.

From Editor To Assembler

After you are sure that you've correctly entered the program, save it to disk (or tape on Atari, C-64, or TI). To save your file, select option (2) Files. Then select the appropriate menu options, and enter the file name. If your operating system does not normally support extensions to file names (all but Atari and IBM), the name must be at least two characters shorter than a normal legal file name. The program will automatically append a .S (_S on the TI), for Source, so that you can use the same name for both source and object files without any confusion. If you have a printer, you may also wish to get a hardcopy of your program. This is helpful when you are tracking down errors during assembly. To use

CONTROL CAPSULE

NanoEditor

KEY	FUNCTION
ESC	Escape
Edit Mode:	
BACKSPACE	Backspace
CONTROL D	Erase line
TAB	Tab.
←	Cursor left
→	Cursor right
RETURN	Enter Line

CONTROL CAPSULE

NanoEditor

KEY	FUNCTION
ESC	Escape
Edit Mode:	
DELETE	Backspace
SHIFT DELETE	Erase line
TAB	Tab
CONTROL ←	Cursor left
CONTROL →	Cursor right
RETURN	Enter Line

CONTROL CAPSULE

NanoEditor

KEY	FUNCTION
F1	Escape
Edit Mode:	
DEL	Backspace
F3	Erase line
F5	Tab
CRSR ←	Cursor left
CRSR →	Cursor right
RETURN	Enter line

CONTROL CAPSULE

NanoEditor

KEY	FUNCTION
ESCAPE	Escape
Edit Mode:	
BACKSPACE	Backspace
DELETE	Delete character
TAB	Tab
←	Cursor left.
→	Cursor right
ENTER	Enter line

CONTROL CAPSULE

NanoEdtior

KEY	FUNCTION
FTCN 9	Escape
Edit Mode:	
FCTN 1	Delete
FCTN 3	Erase line
FCTN 7	Tab.
FCTN S	Cursor left
FCTN D	Cursor right
ENTER	Enter line

the Print option, just select it from the main menu (3). After you save (and print) the source file, select the Exit option from the main menu. The program gives you a chance to change your mind before ending, so you don't need to worry about losing the program in memory due to an erroneous keypress.

Now it is time to load and RUN the *NanoAssembler*. The program prompts you to load your source file for assembly. As the program translates your source code into machine code, it lists the source file, the addresses, and object code to either the screen or a printer (if you have one).

Passing Through

The actual assembly of the program occurs in two steps, or "passes." Thus, the *NanoAssembler* is a "two-pass" assembler. The first pass does most of the work, determining the correct machine-language instructions and the instruction addresses. However, sorting out labels requires a second pass because, until it identifies all address labels, the program may not know the exact address of each instruction.

Try assembling Sample Program 1. If you have entered it correctly, the *NanoAssembler* should output the assembled version, as shown in Figure 1, to the screen or printer. If you have made an error in entering the program into the *NanoEditor*, the *NanoAssembler* informs you of the line number in the source code that contains the error, and states the type of error. For example, if in line 1 you enter LDA #3 instead of LDA# 3, when you try to assemble the program the computer displays the error: ILLEGAL USE OF LABEL IN LINE 1. Here, the computer interprets the code as a LoaD A *addr* instruction (object code = 2), instead of a LoaD A immediate instruction (object code = 1). Then, when the computer evaluates the "label" #3, it finds that the label is illegal because it does not begin with a letter.

Figure 1: Instruction Set

Dec.	Binary	Nibbles per instr.	Mnemonic	Flags* affected C Z	Function
0	%0000	1	ADD	Y Y	Add the contents of B register to the contents of A register—result in A.
1	%0001	2	LDA#	N Y	Load A with number following instruction.
2	%0010	3	LDA *addr*	N Y	Load A with number at location specified by *addr*.
3	%0011	3	STA *addr*	N N	Store the contents of A at location specified by *addr*.
4	%0100	1	TAB	N N	Transfer contents of A to B.
5	%0101	1	TBA	N Y	Transfer contents of B to A.
6	%0110	1	RRC	Y Y	Rotate A right one bit through carry.
7	%0111	1	RLC	Y Y	Rotate A left one bit through carry.
8	%1000	1	AND	Y Y	Logically AND A and B—Result in A.
9	%1001	1	OR	Y Y	Logically OR A and B—Result in A.
10	%1010	1	XOR	Y Y	Logically XOR A and B—Result in A.
11	%1011	3	BZ *addr*	N N	Branch to *addr* if Zero flag is set.
12	%1100	3	BNZ *addr*	N N	Branch to *addr* if Zero flag is not set.
13	%1101	3	BCS *addr*	N N	Branch to *addr* if Carry flag is set.
14	%1110	3	BCC *addr*	N N	Branch to *addr* if Carry flag is not set.
15	%1111	3	JMP *addr*	N N	Branch to *addr* unconditionally.
Assembler Directives:					
n/a	n/a	0	ORG	n/a	Use to specify a particular address (e.g., specify starting address of program).
n/a	n/a	0	EQU	n/a	Equate label with value—assigns the value to the right of the EQU statement to the label to the left.
n/a	n/a	1	DN	n/a	Define Nibble—assigns the value to the right of the DN statement to the label at the left.

**Flags affected refers to whether or not the instruction has any effect on the flags in the status register. The C column stands for the Carry flag (did the operation result in a carry being generated?), and the Z stands for the Zero flag (did the operation result in a zero?). A Y appears in the column if the flag is affected by the instruction. An N indicates the flag is not changed by the instruction.*

After displaying the program, *NanoAssembler* prompts you to save the object file. The saved file is identical in format to the ones you loaded and saved with the *NanoProcessor* last issue; that is, the file contains the contents of all addresses from 0 through 255. To see that your program works properly, load and RUN the *NanoProcessor*. You can then load and run the program you've just created according to the instructions detailed in Vol. 5, No. 5.

For a short program such as Sample Program 1, this process may seem a bit time consuming. For longer and more complex programs, however, the ease of writing and debugging provided by an assembler more than makes up for the added steps.

Assembler Directives

Figure 1 displays the 16 instructions that we detailed in the *NanoProcessor*. You may specify any of these instructions when writing an assembly-language program with the *NanoEditor*. The *NanoAssembler*, in turn, converts these instructions into their machine codes. There are three additional commands, known as assembler directives, that the *Assembler* understands:

Directive	Purpose
ORG	Start object code here
DN	Define a nibble
EQU	Define a label

The ORG command directs the *NanoAssembler* to assemble the program at a specified address between 0 and 255. For an example of this instruction, see line 1 of Sample Program 2. This program is a slightly modified version of Sample Program 2 that we presented in last issue's *NanoProcessor*. It performs a two nibble addition of numbers located at addresses 240 and 241, placing the answer in addresses 248 and 249. The ORG statement makes the starting address %1010.

The DN instruction allows you to include a particular value at any address. Just specify the address using the ORG directive, and then define the value to be placed at that address with the DN directive. Lines 22 through 24 of Sample Program 2 define the two nibbles that the program adds.

Figure 2

Decimal	Binary	Hexadecimal
0	%0000	$0
1	%0001	$1
2	%0010	$2
3	%0011	$3
4	%0100	$4
5	%0101	$5
6	%0110	$6
7	%0111	$7
8	%1000	$8
9	%1001	$9
10	%1010	$A
11	%1011	$B
12	%1100	$C
13	%1101	$D
14	%1110	$E
15	%1111	$F

Sample Program 1

Addr	Code	Line	Label	Mnemonic	Remark
		001		LDA# 3	;Get first number
0	%0001				
1	%0011				
		002		TAB	;Move to B
2	%0100				
		003		LDA# 7	;Get second number
3	%0001				
4	%0111				
		004		ADD	;Figure sum
5	%0000				
		005	HERE	JMP HERE	;Jump self to stop
6	%1111				
7	%0110				
8	%0000				

The EQU command lets you identify any address with a particular label. Lines 2 through 6 of Sample Program 2 use this directive. These statements make Sample Program 2 more readable by assigning descriptive labels to the 5 data addresses: NIB1 and NIB2 for the two numbers to be added; LONIB and HINIB for the low and high nibbles of the answer; and OUT for the OUT light. (See last issue's *NanoProcessor* for a complete explanation of how Program 2 uses these 4 locations.)

The other change to Program 2 in this issue is in the use of the OUT light located at the upper-left of the *NanoProcessor* screen. When you assemble Sample Program 2 and run it, you will find that the OUT light is off when the program begins, but it turns on when the program is complete. Thus, you do not need to know what address the program will end on. Instead, the OUT light signals that the program is finished.

Sample Program 3 accesses the *NanoProcessor*'s "sound chip." Any time you store a number at either location 254 or 255, the *NanoProcessor* responds with a tone. With 16 different values possible at each of these locations, you can make a total of 32 different tones. Sample Program 3 plays a C scale.

We hope that you have found these *Nano* programs instructive and enjoyable. With what you have learned, you should be able to create your own "machine-language" routines. Feel free to let us know in "Letters to the Editor" of any programs you create, so we may share them with our readers.

HCM Glossary Terms: assembler, label, object code, op-code, operand, pass, source code.

For your type-in listings, see HCM PROGRAM LISTING CONTENTS.

HCM

Three Number Systems Supported

Machine language on the *NanoProcessor* can be entered only in binary. The *NanoAssembler*, however, understands decimal and hexadecimal in addition to binary. Last issue we explained how to convert between decimal and binary—this issue we introduce you to hexadecimal.

As we explained in the previous issue, decimal is a base 10 system. It uses ten digits (0 through 9) to represent numbers. Similarly, binary is a base 2 system and uses two digits (0 and 1). Hexadecimal is a base 16 number system and uses 16 different digits—0 through 9 plus the letters A through F. (See Figure 2 for a conversion chart.) As the conversion chart shows, we can express the number 11 decimal as either the binary number %1010 or the hexadecimal number $B. (Note that the % symbol denotes a binary number, and the $ symbol a hexadecimal number.)

To convert a two-digit hexadecimal number (say $C8) to decimal you simply find the decimal equivalent of the left-most digit (i.e., $C = 12), and multiply it by 16. Then simply add the decimal equivalent of the right-most digit ($12*16+8=200$). Hexadecimal is a particularly useful system in assembly language because it can express any nibble as a single character or any byte as two characters.

Sample Program 2

```
Addr  Code    Line  Label  Mnemonic   Remark
                001          ORG 10
                002  NIB1    EQU $F0
                003  NIB2    EQU $F1
                004  LONIB   EQU $F8
                005  HINIB   EQU $F9
                006  OUT     EQU $FD
                007          LDA# 0     ;Turn OUT light off
 10   %0001
 11   %0000
                008          STA OUT
 12   %0011
 13   %1101
 14   %1111
                009          LDA NIB1   ;Get first number
 15   %0010
 16   %0000
 17   %1111
                010          TAB        ;Move to B
 18   %0100
                011          LDA NIB2   ;Get second number
 19   %0010
 20   %0001
 21   %1111
                012          ADD        ;Figure sum
 22   %0000
                013          STA LONIB  ;Low to memory
 23   %0011
 24   %1000
 25   %1111
                014          BCC NIB    ;One nibble answer
 26   %1110
 27   %0010
 28   %0010
                015          LDA# 1
 29   %0001
 30   %0001
                016          JMP STHI   ;All done
 31   %1111
 32   %0100
 33   %0010
                017  NIB     LDA# 0     ;Zero A
 34   %0001
 35   %0000
                018  STHI    STA HINIB  ;High to memory
 36   %0011
 37   %1001
 38   %1111
                019          LDA# ON    ;Set OUT light
 39   %0001
 40   %0001
                020          STA OUT
 41   %0011
 42   %1101
 43   %1111
                021  HERE    JMP HERE   ;Jump self to end
 44   %1111
 45   %1100
 46   %0010
                022          ORG $F0
                023          DN $A
                024          DN $C
```

Sample Program 3

```
Addr  Code    Line  Label  Mnemonic   Remark
                001  SOUND   EQU 254
                002          LDA# 2
 0    %0001
 1    %0010
                003          TAB
 2    %0100
                004          AND
 3    %1000
                005          RRC
 4    %0110
                006          STA SOUND
 5    %0011
 6    %1110
 7    %1111
                007          ADD
 8    %0000
                008          STA SOUND
 9    %0011
 10   %1110
 11   %1111
                009          ADD
 12   %0000
                010          STA SOUND
 13   %0011
 14   %1110
 15   %1111
                011          LDA# 6
 16   %0001
 17   %0110
                012          STA SOUND
 18   %0011
 19   %1110
 20   %1111
                013          ADD
 21   %0000
                014          STA SOUND
 22   %0011
 23   %1110
 24   %1111
                015          ADD
 25   %0000
                016          STA SOUND
 26   %0011
 27   %1110
 28   %1111
                017          ADD
 29   %0000
                018          STA SOUND
 30   %0011
 31   %1110
 32   %1111
                019          LDA# $D
 33   %0001
 34   %1101
                020          STA SOUND
 35   %1101
 36   %1110
 37   %1111
                021  HERE    JMP HERE
 38   %1111
 39   %0110
 40   %0010
```

HCM Review Criteria

Each month, *Home Computer Magazine* (*HCM*) reviews products designed for the Apple II family, Atari 800 family and compatibles, Commodore 64, IBM PC and PCjr, and Texas Instruments 99/4A computers. *HCM* reviews take a detailed look at the quality, utility, and value of commercially available packages for these machines. Because our publishing charter forbids accepting outside advertising, we strive to make the scope and content of our review pages shine with a unique blend of humanistic frankness and objectivity.

Not only will you find all relevant information for making a wise purchase decision, but in some special cases we also provide nuggets of compu-prestidigitation.* For example, we frequently include essential documentation not furnished by the manufacturer. Additionally, each issue of *HCM* tries to review at least one outstanding product—a "Diamond in the Rough"—which, because of company size, marketing clout, or for some other reason, has not received the attention it deserves.

At the beginning of each review, a review-at-a-glance box provides the user with an instant assessment of the product. Each item will be evaluated, where relevant, with the criteria below.

HCM Review



Name: Old Art
Program Type: Recycled Graphics
Machine: Apple II family, Atari 800XL, C-64 IBM PC & PCjr, TI-99/4A
Distributor: Hit 'n' RUN Software, Inc.
Price: $99.99 (or trade for '72 Pinto)
System Requirements: Disk Drive, Joystick, Trash Can optional

	Poor	Fair	Good	Excellent
Performance:				
Engrossment:				
Documentation:				

Products may also be evaluated in the following areas:

*** Flexibility—**
Can the product be adapted to the specific needs of the users?

*** Cost/Benefit—**
Is the product worth the user's investment in time and money?

*** Necessity—**
Is the product a solution for which a problem already exists?

*** Originality—**
Is it unique in concept, or simply a "me too" product?

*** Longevity—**
The "Boredom Factor." Does the program sustain interest?

*** Rewards—**
Are the audio-visual rewards motivating and appropriate?

*** Concept Presentation—**
Are the concepts presented clearly, logically, and in depth?

*** Special Effects—**
How does quality of sound and visual effects rate? Do they enhance or detract from the product or learning process?

*** Performance—**
How well the product performs as intended; how well it takes advantage of a specific machine's capabilities; how well it responds to the user's commands; how effectively the graphics, sound effects, music, or speech are integrated with the software.

*** Engrossment—**
Whether the game or activity has that intangible quality that holds players on the edge of their seats while the hours tick by unnoticed.

OR

*** Ease of Use—**
The degree to which a user can interact with the product without outside help: the ease and effectiveness of error-handling features; whether the actual reading level of the activity is appropriate for the suggested audience.

OR

*** Ease of Set-up—**
How well the product design facilitates easy installation.

*** Documentation—**
The quality of the printed matter that comes with the product; whether the instructions are clear and comprehensive; whether the machine configuration requirements are spelled out. Information such as how to load a program, use the keyboard, and restart an activity contributes to the documentation rating, as do tips on performance peculiarities.

Attention Software Authors & Peripheral Inventors:

* WANT TO BE DISCOVERED? *

Home Computer Magazine Wants To Give You A Chance!

We are looking for home computer products that have not received the attention they deserve. Each month, we will be singling out one such package for special review. If you have a unique commercial product of exceptional quality—but your advertising and promotion budget has not allowed you to capture major media attention—we want to see it. We will consider reviewing any product that meets our high standards.

We are an Equal Opportunity Reviewer!

In order to qualify for possible review, your product must:

1. Currently be available for purchase to readers of this magazine.
2. Make a unique and important contribution to the home computer industry.
3. Be of outstanding merit, quality, and value.
4. Be consistent with the type of machines and products we normally cover.

If you feel that your product qualifies, mail it to:

Home Computer Magazine
Attn: Editorial Submissions
1500 Valley River Drive, Suite 250
Eugene, OR. 97401

We reserve the right *not* to reply to each inquiry, so please do *not* contact us except to request return of your product. If you want your product to be returned, please include sufficient return postage.

***Compu-prestidigitation** (kóm´•pū•prĕs´•teh•dĭ•jeh•tā´•shŭn) —*n* 1. The magical quality of unexpected comprehension that results from presenting technical information about computers in a lively, entertaining, visually attractive and easy-to-understand format. 2. The magical tricks that make a computer sing, dance, and do all sorts of wonderfully useful things.

JUNIOR ENHANCEMENTS

by Laile L. Di Silvestro
HCM Staff

Do you yearn for the memory of an elephant, the speed of a mouse? Perhaps these two PCjr peripherals can help.

How many IBM PCjr owners haven't yearned for the expansive memory of Junior's larger brothers? For a second disk drive? How many haven't sighed when passing up memory-hungry programs such as *Volkswriter*, *Framework*, and *Multiplan*? Well, the market bears good news: With the help of a peripheral or two, Junior can grow.

Several memory expansion systems are available, providing the PCjr with additional memory ranging from 64K to 384K. Most systems include one or more extra features that the Junior normally lacks, such as a second disk drive, a parallel port, and a clock/calendar. Such systems promise to provide the speed and flexibility associated with the IBM PC.

In previous issues, we reviewed three expansion systems: the *Quadjr Expansion Chassis*, the *Legacy II*, and the *Tecmar jrCaptain*. [See "Romancing the PCjr: A Review of the Quadjr Expansion Chassis" in Vol.5, No.4, "Legacy II for the PCjr" in Vol.5, No.2, and "Jr. Addition: A Review of the Tecmar jrCaptain Peripheral" in Vol.4, No.4—Ed.] In this issue we review two more—the *Microsoft Booster* and the *IBM PCjr Memory Expansion Attachment*. Both systems provide an additional 128K, RAM drive, and software utilities intended to satisfy the memory-starved PCjr owner's every craving. Because the documentation that accompanies both of these systems lacks essential information, in this review we pass some of the missing pieces on to you.

Moving Memory

Junior presents a challenge to those who would expand its memory. The difficulty lies in Junior's unique memory allocation. Unlike the IBM PC, which has dedicated Random Access Memory (RAM) for video memory (located at $B8000 in high memory address space—the dollar sign denotes hexadecimal), Junior's video buffer is located in low memory address space in the middle of system RAM. When powering up, the IBM Expanded PCjr expects to find this video buffer in the same place each time: the last 16K (locations $16580 through $1F400) of its 128K RAM. Once Junior's Disk Operating System (DOS) locates the video buffer, it ceases its search for memory. Therefore, it cannot access any memory added beyond this video buffer.

Memory expansion systems overcome this problem by using software to move the video buffer into the DOS section of memory. As a result, the memory available for application programs opens up, thereby becoming contiguous with the additional memory. The program responsible for this rearrangement is activated through a CONFIG.SYS file—a file that runs as DOS loads into memory, reconfiguring the way in which Junior views its memory.

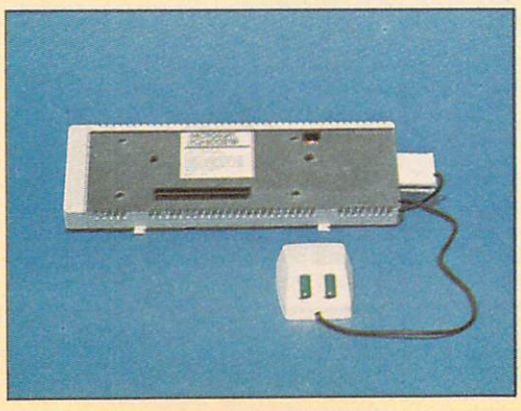
You can also use the CONFIG.SYS file to activate a simulated disk drive known as RAM drive. You can do almost everything with RAM drive that you would do with your normal disk drive: copy files into the drive, save files, and load files into system memory or onto the screen. Because the methods involved in creating a RAM drive differ with each product, we will discuss them in detail in separate sections below.

The main advantage of RAM drive is its operating speed. Because they are not hampered by mechanical devices or serial processing, most RAM drives operate much faster than conventional disk drives. The major disadvantage of RAM drive is that a system reboot or a power cutoff results in complete data loss.

I. The Microsoft Booster

Microsoft's *PCjr Booster* is a 1.5 watt circuit module that provides your Junior with an additional 128K. It is equipped with a port intended for use with Microsoft's mouse. [Watch for a review of this product in our "rodent round-up" next issue—Ed.] Do you usually skip entering the date and time when you power up Junior? The *Microsoft Booster* provides a battery-powered clock that will take care of the task for you. The package includes the *Microsoft Mouse*, software, the Installation and Operation Manual, a Microsoft Mouse Menu manual, and a 4.5 Volt battery.

Installation of the circuit module couldn't be simpler. You have only to pop the side cover off of the I/O expansion port, press the module into place, attach it with the provided screws, and re-install the side cover. To increase Junior's memory up to 512K, you can piggy back additional modules onto the original *Booster*, using the



Name:	Microsoft Booster
Product Type:	Hardware/software memory expansion
Machines:	IBM PCjr
Distributor:	Microsoft Corporation 10700 Northrup Way Bellvue, WA 99004 (206) 828-8080

	Poor	Fair	Good	Excellent
Performance:			■	
Ease of Use:			■	
Ease of Setup:			■	
Documentation:		■		
Cost/Benefit:	■			

same installation procedure. Due to Junior's limited power supply, however, you must install the *IBM Power Expansion Attachment* to accomodate the extra power needs of additional modules.

Next, you must set the memory address switches. These three switches (located on the side of the module) serve to inform Junior of the location of the additional memory.

With the *Booster* installed, all that remains is to reconfigure the way that Junior views its memory, so that it can access the additional memory. Microsoft software provides CONFIG.SYS files that do the job. CONFIG1.SYS and CONFIG2.SYS are sample files that you may use for different applications: The first provides the maximum amount of application-usable memory and is best suited for programs that require large amounts of memory; the second creates a RAM drive. Both files contain the following command lines, which set up the mouse and clock functions:

```
DEVICE = CLOCK.SYS
DEVICE = MOUSE.SYS
```

In addition, CONFIG2.SYS contains a line that establishes the memory configuration: DEVICE = MEMORY.SYS. This must be the first command in CONFIG.SYS.

You can pass three parameters to MEMORY.SYS via the CONFIG.SYS file. /s:*nn* establishes the size of RAM drive (*nn* = a number between 1 and 80 representing 6K to 91K); /d:*nn* determines the number of files that you can store in RAM drive (*nn* = a number between 4 and 64); /v:*nn* specifies the size of the video buffer (*nn* = 2K to 96K in 2K increments). The easiest way to set these parameters is to use the DOS EDLIN.COM program to edit CONFIG.SYS to alter the parameters passed to MEMORY.SYS.

MEMORY.SYS stores RAM drive with DOS and the video buffer in Junior's internal memory, rather than in the additional memory. If you specify the maximum amount of memory (80), RAM drive receives the memory that is left over after DOS and the video buffer are installed. Therefore the maximum amount of memory RAM drive can hold is 91K. [Microsoft representatives claim that a more recent version of the *Booster* software allows you to create a RAM drive with a capacity of 256K minus room for DOS, the device driver, and the application—Ed.] It is impossible to specify the exact size of RAM drive. For one thing, the numbers (1-80) that determine its size are relative figures that do not correspond to actual amounts of memory. The exact amount depends on how much memory DOS and the video buffer appropriate. In addition, the manner in which Junior stores its video buffer sometimes ties up additional blocks of memory. To determine the exact size of RAM drive, use the CHKDSK.COM program (described in the IBM DOS manual).

The Installation and Operation Manual seems to be aimed at the more experienced computer user; well-written notes and charts convey side information that can both benefit and interest those that want to know the reason behind the procedures. The manual's main content, however, is scanty and imprecise. For example, to instruct you on the setting of switches, the documentation directs you to a misleading illustration of a different type of switch than the module is actually equipped with. To determine the correct switch settings, you must ignore the text and refer to a chart that was intended only for multiple-module installation. For a first-time user, such inconsistency is confusing.

II. IBM Memory Expansion Attachment

IBM's memory expansion system is quite similar to the *Microsoft Booster*: It provides an additional 128K of memory; it comes with a software disk to reconfigure Junior's memory; and it attaches to Junior's I/O expansion port. The IBM expansion differs in that it offers no extra features such as mouse, clock, or device port. In addition, the IBM expansion requires that you use it in conjunction with the *IBM Power Expansion Attachment*.

> ***"You could be pleasantly surprised at the enhanced abilities that these systems can provide—but only after considerable labor and tedious periods of trial and error."***

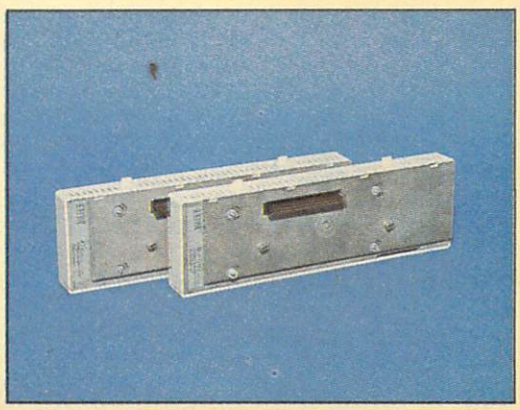
The CONFIG.SYS file must contain PCJRMEM.COM to reconfigure memory properly. PCJRMEM.COM relocates the video buffer and determines its size. If you include RAMDISK.COM, you also create a RAM drive. You can install these files by using the Automatic Installation program. This program is lengthy, however, and you may prefer to install the files yourself using the EDLIN and COPY commands described in the IBM DOS manual.

In terms of video buffer size, IBM's software does not allow you as much flexibility as Microsoft's—you have only three choices: 16K, 32K, and 96K. IBM offers considerable flexibility in RAM drive size, however. Because the IBM software does not consign RAM drive to system memory only, your sole limitation is the room that you must leave for DOS, the device driver, and the application. Unfortunately, it is difficult to determine exactly how large a RAM drive an application can tolerate, and the manual provides no help. Your only recourse is time-consuming experimentation.

The manual constitutes the worst memory-expansion documentation we have encountered so far. What information the manual contains is superfluous and redundant; and the most vital information is lacking.

Name:	IBM Memory Expansion Attachment
Product Type:	Hardware/software memory expansion
Machines:	IBM PCjr
Distributor:	IBM Corporation 900 King St. Ryebrook, NY 10573 (800) 426-2468

	Poor	Fair	Good	Excellent
Performance:			■	
Ease of Use:		■		
Ease of Setup:				■
Documentation:	■			
Cost/Benefit:		■		



STOP THE PRESS!

As this issue was going to press, Microsoft informed us that their *Booster* has been discontinued. Microsoft also claimed that a new version was issued before the product was discontinued. This version reportedly allows a greater RAM-drive capacity. However, Microsoft representives would not give us any further information about the availability of the booster, its discontinuance date, or how the new version can be distinguished from the old.

FIGURE 1 A comparison of three similar memory expansion systems. See *HCM* Vol.4, No.4 for a review of the Tecmar jrCaptain.

PRODUCT	RAM DISK CAPACITY	CLOCK	MOUSE PORT	PARALLEL PORT	REQUIREMENTS	PRICE
Microsoft Booster with mouse	91K	YES	YES	NO	Expanded PCjr DOS 2.1	$495
IBM Memory Expansion	256K minus room for device driver, DOS, applications	NO	NO	NO	Expanded PCjr Power Expansion DOS 2.1	$325 plus $150 for Power Expansion
Tecmar jrCaptain	256K minus room for device driver, DOS, applications	YES	NO	YES	Expanded PCjr DOS 2.1	$395

For example, the manual explains how to make a backup disk *three times*, yet it neglects to describe the procedure for accessing PCJRMEM.COM and RAMDISK.COM in order to specify the size of RAM drive and the video buffer. Fortunately, the software disk includes a PCJRMEM.DOC file that contains the Technical Reference Manual—a fact not revealed by the operating instructions. This manual is somewhat better than the operating instructions and is absolutely essential if you want to to manipulate RAM drive.

The Elusive RAM

You could be pleasantly surprised at the enhanced abilities that the Microsoft and IBM memory expansion systems can give the PCjr—but only after you have undergone considerable labor and tedious periods of trial and error.

The RAM drive potential to eliminate the hassle of a single drive, for example, is exciting. We imagined storing entire applications in RAM drive so that the disk drive would be free for a data disk or a second systems disk. In this way, we could conceivably use programs such as *Easywriter* that require heavy use of two disks (see our review of this word-processor in *HCM* Vol.4, No. 3) without the bother of constantly switching disks. To our disappointment, however, the RAM drive memory (91K with the *Microsoft Booster*; 256 minus room for DOS, the device driver, and any application with the *IBM Memory Expansion*) is rarely sufficient for an entire application—a debilitating limitation. Although programmers might find the extra memory beneficial, we have found no PC software for which such a small amount of memory is adequate. (See the Counterpoint for a programmer's point of view.) In addition, some programs are hard-coded to drives A and B, so they cannot access a third, RAM drive.

Given RAM drive's limited capacity, it performs best if used for the temporary storage of data. Unfortunately, you must copy all data onto a disk before you turn the power off or reboot—otherwise your work is for naught. Of course, some of the time that you save by using a RAM drive is consumed when you copy files into RAM drive and then onto a disk when you finish working with your computer. But time factors and the fear of data loss put aside, using a RAM drive can be a delight. Transferring data from screen to RAM drive and back takes half the time it would if you used a traditional disk drive. And—especially if you find an application which fits into the simulated drive—the novelty and joy of not having to switch disks can mitigate any negative features.

Considering the expense of beefing up Junior to match the PC's capabilities and the difficulties involved in memory expansion operation, it might be more worthwhile to simply buy the PC itself. Yet, if you already own a Junior and want to use programs normally restricted to the PC, memory expansion systems could be for you.

COUNTERPOINT

For a programmer, beefing up the memory of a PCjr is an attractive prospect. Even though IBM BASIC is limited to 64K of RAM, a RAM drive can help shave down development time. By placing certain tried-and-true routines in RAM drive at the beginning of a session, you can access those files in the wink of an eye. Both the *Microsoft Booster* and the *IBM Memory Attachment* make RAM drive available. The disk-based documentation included with the IBM product is very informative; however, it is much easier to find the best memory configuration with the Microsoft unit—due to its slightly better documentation. With the IBM software, it is difficult for the BASIC programmer to determine the best location and size of the video buffer.

I have to give the *Microsoft Booster* a slight edge as a programming tool, because its accompanying software includes a file called JBASIC.COM, which loads PCjr Cartridge BASIC into the memory expansion instead of into Junior's internal RAM. Our tests show that BASIC works perfectly with the *Microsoft Booster* and RUNs a good 20 percent faster than normal. Although IBM's memory expansion can theoretically give you a larger RAM drive than the *Microsoft Booster*, I found that trying to access the higher resolution screens (i.e., SCREEN 5) causes the IBM system to behave erratically, and even lock up—regardless of which parameters I pass to PCJRMEM.COM .

—Roger Wood

HCM Glossary terms: CONFIG.SYS, DOS, hard-code, hexadecimal, parameter passing, RAM, video buffer.

HCM

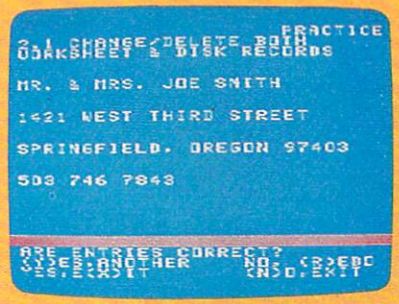


MOVING?
Don't Miss Out On Any Issues Of
HOME COMPUTER magazine

Send us a Change-of-Address Card (available at any Post Office) 6-8 weeks prior to the move.
Be sure to include both the old & new address, plus the alphanumeric code above your name on the mailing label.

Please send this information to:
Home Computer Magazine
P.O. Box 70288
Eugene, OR 97401

HCM Review



Name:	File Directory
Program Type:	Utility
Machines:	TI-99/4A
Distributor:	John Bulakowski 162 Lorann Drive Naugatuck, CT 06770
Price:	$29.95
System Requirements:	32K memory expansion, disk drive

	Poor	Fair	Good	Excellent
Performance:				
Ease of Use:				
Documentation:				
Cost/Benefit:				

FILE DIRECTORY

A Review by Steve Nelson

HCM Staff

This program won't turn your house into a post office, but it could make your monthly mailing chores a little bit easier.

John Bulakowski's *File Directory* is a single-purpose program, designed simply to store, access, and print mailing labels. Small, narrowly-focused programs can, by their very simplicity, make specific tasks easier—eliminating unnecessary distractions. But, in its simplicity, can *File Directory* enable you to perform your mailroom chores with ease?

File Directory requires a disk drive, Extended Basic, and 32K memory expansion. A single disk is capable of maintaining 6 files, each with up to 100 records. Each record consists of 4 fields: name; street address; city, state, and zip code; and telephone number.

Entering Records

File Directory boots automatically when you select the Extended BASIC option. Once you select a file to work on, you can create a new record or change existing records. To create a new record, enter all of the data (name, address, etc.) into the 4 fields. The program automatically saves the information for you when you exit the program or begin working on another record. The program can also list all of a file's records—or just the names—either to the screen or to a printer.

File Directory lets you alter selected portions of your records. Using the [FCTN D] or [FCTN S] options, you can change a single word or letter without altering any other part of the record. Unfortunately, the first time you try to delete an entire record, you may experience some difficulty because there is a discrepancy between the instructions in the documentation and the prompt on screen. The on-screen prompt directs you to DELETE RECORD WITH ERASE. The documentation reads "If you want to delete this record, enter FCTN 1 (DEL) and input ENTER." This is an obvious error. You *can* delete the record using [FCTN 1], but to do so, you must hold the key down until it deletes all of the letters in the top line of the first field, then press [ENTER]. On the other hand, the erase function on the TI-99/4A—[FCTN 3]—deletes files immediately and with ease.

This error detracts from the overall quality of the documentation, which, though extensive, is sometimes poorly-worded and often confusing.

Printing Possibilities

File Directory prints labels and envelopes to specific preset tab locations, although you can manipulate them if you need to. *File Directory* is written for use with a parallel printer. But if you are using a serial printer, the documentation provides instructions on editing the program to allow you to use it. The program works with any printer that is compatible with the TI-99/4A computer and that can recognize the program's imbedded commands.

The program contains two printing options: Exception and Selection. Printing by Exception allows you to receive a printout in three possible typestyles: correspondence quality, standard, or expanded. Correspondence quality is crisper than standard quality, and expanded is a blown-up version of the typestyle used with correspondence quality. In the Exception mode, *File Directory* prints all files that you do not "except." After you select the option, the program prompts you for deletions. Use the [FCTN 3] option to delete the records that you don't want printed, and select a typestyle. The program then prints out all but the records you have excepted, without deleting any records from disk. You can print an entire file simply by making no "exceptions."

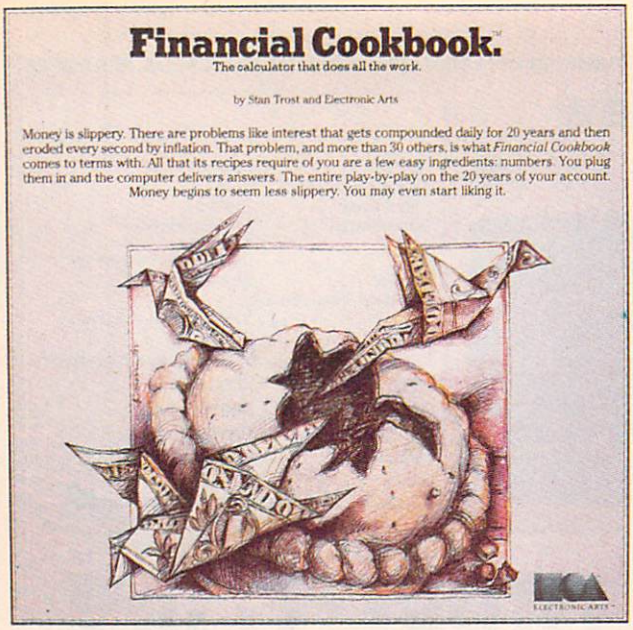
The Selection option offers you only two typestyles: correspondence quality and standard. The program prompts you for two record numbers, then prints the records you specify. Each time it completes printing, the program prompts you for two more numbers and repeats the procedure until you finish. Unfortunately, this printing method has the unnecessary drawback of tying you to the keyboard.

You can find phone numbers with ease, using the phone-number option. Enter the name (or partial name) of the person whose phone number you want, and *File Directory* searches through the records and prints the number or displays it on the screen.

Finally, *File Directory* has the handy feature of sorting your records alphabetically by last name. The program sorts by looking for the space before the last name in each record. This means that if you enter a name like De Silva and leave a space before Silva, the program sorts Silva, not De Silva. The sorting process is also somewhat slow, to say the least—taking about 30 minutes for 100 records!

Considering *File Directory*'s poor documentation and extrememly slow response time, many TI-99/4A owners might prefer their old address-filing method—scratching and erasing in an old, worn-out address book. But for those of you with more than a little time and patience to spare, *File Directory* is a safe, tidy way to organize and print all of your important addresses.

HCM



Financial Cookbook

A Review by Michael Hosmar
HCM Staff

Use these money-managing recipes to sauté some tasty financial advice.

Some of Grandma's best recipes contain very few ingredients. She always says that the simplest things in life are sometimes the best.

Financial Cookbook, a straight-forward money manager from Electronic Arts, was built on the same premise. The program offers 32 recipes that answer a myriad of such financial questions as: Can I afford to buy a new car? Should I refinance my mortgage? Is an IRA the best way for me to save money for retirement?

You can't answer these questions, however, without looking at rates for taxes, interest, and inflation. And if you're like me, these concepts seem much more complicated than one of Grandma's recipes. But *Financial Cookbook* makes them easy to understand and takes your mind off the calculations so you can concentrate on the results.

The *Cookbook*'s recipes can cure you of "number phobia," while making you more confident about your financial situation. The program lists the ingredients—all you have to do is watch them cook.

Financial Cookbook is extremely easy to set up and can be used with the Apple II family, Macintosh, Commodore 64, IBM, and Atari computers. Within seconds, the program loads, and the Index of recipes appears on the screen. (The Commodore version is the exception here, taking over three minutes to load—a function of the slow Commodore disk drive.)

> ***"The Cookbook's recipes can cure you of 'number phobia,' while making you more confident about your financial situation."***

Financial Entrees

Using *Financial Cookbook* is a breeze—nothing like trying to steal Grandma's secret for apple pie. You have a choice of 32 very accessible and easy-to-understand recipes. They help you discover the "ins and outs" of interest rates, mortgages, financing, refinancing, life insurance, and much more.

Press only one key, and the list of the 32 formulas scrolls down the screen. Simply place the cursor on the recipe you want to use, press one more key, and you're ready to start plugging in numbers.

Let's say you want to know how much money you'll need to deposit in your savings account each month to purchase a new car. Select MONTHLY DEPOSIT FOR FUTURE PURCHASE in the index of recipes. When the recipe appears, type in the appropriate numbers and tell the computer to make the calculations. When the answers flash onto the screen, move the cursor down to scroll through the monthly and yearly figures. Try different inputs, and compare the results.

One of the program's most useful functions—Profile—inserts specified values automatically whenever a recipe calls for them. In other words, if you want to hold your marginal tax rate constant, type it into the Profile function. The *Cookbook* then enters this figure into each recipe you use, until you quit the program.

Better Than Betty Crocker

The program's documentation includes more than just operating instructions. *Financial Cookbook*'s conservatively designed package also contains "The Manual"—a 32-page handbook that could double as an entry-level economics course. It contains everything you need to understand the results yielded by this decision-making tool. It also explains a little bit about how each recipe was formulated—something not found even in the best of Betty Crocker's cookbooks.

The Manual's reference-book format makes multiple references easy—each recipe is assigned a number that correlates with a number on the screen. And its 5-page Glossary concisely defines terminology that might be unfamiliar to inexperienced financial chefs.

The Manual's main purpose, however, is to answer questions left unexplained by the operating instructions. Therefore the booklet contains short explanations of each recipe ingredient, such as capital gains rate, marginal tax rate, and pre-tax gain. It also gives you easy-to-understand formulas that you can use to calculate these rates and interprets each result that the computer calculates.

Cookbook Longevity and Flexibility

Grandma's cooking doesn't always please everyone in the family. Similarly, *Financial Cookbook* might not fit into everyone's software kitchen. In fact, some users might think the program is too rigid and limiting. Here's one example: In recipe number 31—BUYING YOUR CAR—there's no way to customize the time inputs for your own needs. The recipe asks how many "years" you plan to keep the car before reselling. But what if you plan to sell it in eight months? Six months? Or in some special cases, only a couple of months? The program won't allow those inputs.

Most *Financial Cookbook* recipes don't need this specificity of time. Savings accounts and IRAs, for example, don't have much value if they last only a few months. But recipes that involve short-term investments *would* contain more utility with this added time dimension.

Another less obvious problem is that you might occasionally enter numbers that the program can't sensibly use. When that happens, a disgruntling message appears on the screen—SORRY, CAN'T HANDLE THOSE INPUTS. Your first response will probably be, "Why not!" The *Cookbook* designers must have overlooked an answer to this question. No prompts flash onto the screen to tell you that your monthly payment input is too low or that you forgot to enter your marginal tax rate. The Manual is no help either. Its explanation of this message merely reiterates the message itself. Therefore, the program makes you hunt down input problems through trial and error.

Low-Risk Investment

Aside from these minor inconveniences, *Financial Cookbook* is a sound investment. All versions of the program perform perfectly. *Financial Cookbook* on the Macintosh is a real treat. The window format and mouse capabilities make entering information a snap. The Mac also lets you simultaneously display a number of results for easy comparison.

Money-management programs are by no means "something new," but *Financial Cookbook* seems to have qualities the others don't. It has found an unoccupied niche in the world of financial decision-making tools.

The program offers more options and answers more questions than most of its competitors. Whereas $200 programs like *Managing Your Money* by Andrew Tobias focus on the intricacies of tax planning, life insurance, and personal budgets, the $40 *Financial Cookbook* gives concrete answers to questions people ask every day: Can I buy a new car? Can we afford our dream home? How can I best save for the future?

This combined calculator and reference source, however, is by no measure a substitute for your professional advisor. Nevertheless, if you're unfamiliar with the way money works, it might help you to get your feet wet in the financial world. It also alerts you to many options available for financial investment, so you can ask the right questions next time you meet with your financial advisor.

So if you're planning to cook something up in the financial world, you might look up the recipes in the *Financial Cookbook*.

COUNTERPOINT

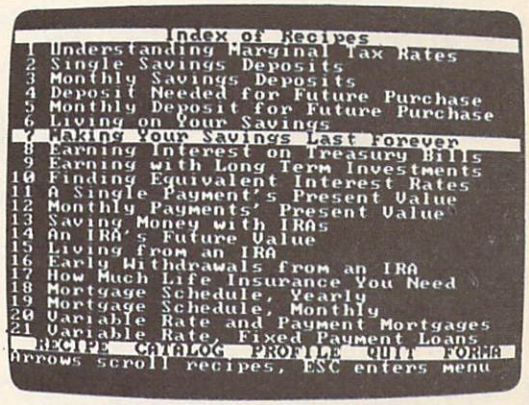
The product's title—*Financial Cookbook*—implies that the software contains a veritable financial "stew" of information. But in reality, the separate programs merely comprise investment and IRA calculations, with a couple of home and auto comparisons thrown in for spice. It looks as if the *Cookbook* designers began with retirement analysis, and then loaded the disk until it was full. Moreover, programs similar to some of these "filler" recipes are readily available in other software packages and calculator resources. *Financial Cookbook* would be better published in volumes—each volume could cover the individual topics more thoroughly.

—Bob Karau

HCM

Name:	Financial Cookbook
Program Type:	Financial decision maker
Machines:	Apple II, II+, IIe, IIc, Macintosh, IBM PC, PCjr, Commodore 64, Atari 800, 800XL, and 130XE
Distributor:	Electronic Arts 2755 Campus Dr. San Mateo, CA 94403 (415) 571-7171
Price:	$40
System Requirements:	Disk drive and 48K

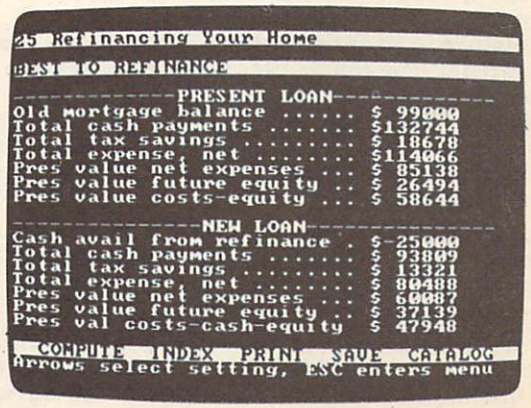
	Poor	Fair	Good	Excellent
Performance:				
Cost/Benefit:				
Ease of Use:				
Documentation:				



Move the cursor down the screen until you find the recipe you want to use.

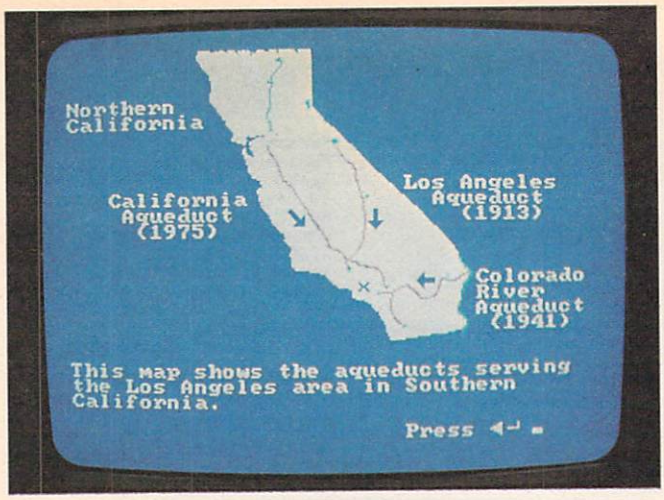


When the recipe appears on the screen, enter information pertaining to your own finances.

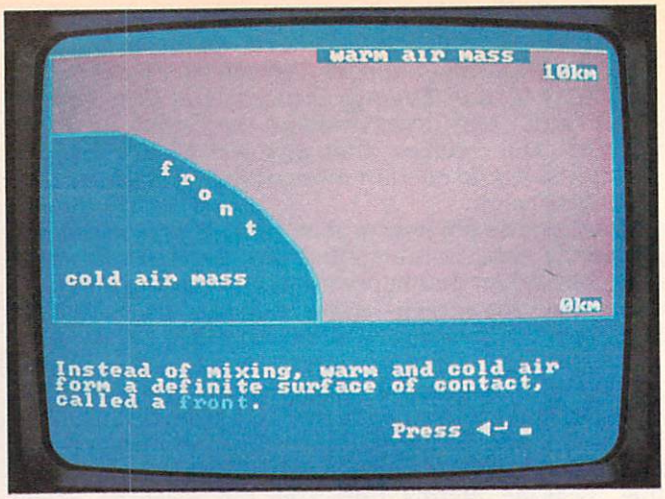


Move the cursor to the bottom of the screen to scroll through the various calculations (e.g., present loan and new loan figures).

PRODUCT REVIEWS



This screen from the Hydrologic Cycle's Human Impact Tutorial shows you the aqueduct system supplying water to Los Angeles.



This screen from Moisture in the Atmosphere's Clouds and Precipitation Tutorial depicts what causes a weather front.

Earth Science Series

A Review by Steve Nelson
HCM Staff

Educators and computers team up to help us study our physical environment . . .but is it an effective way to learn?

Do you know how raindrops form, or how wells and underground springs get their water? Have you ever wondered how a cloud produces snow? With over 70% of the Earth's surface under water, what keeps the rest afloat? If you've ever puzzled over these things, then IBM's *Earth Science Series* is bound to interest you.

The *Earth Science Series* comes in 4 packages, with a fifth package in the works. The programs have excellent graphics and feature an easy-to-use question-and-answer format. Each program is independent of the others and addresses a specific aspect of the physical environment. Here are brief descriptions of the programs:

Hydrologic Cycle—How the sun regulates the cycle of water from the earth to the sky and back to the earth again; the atmosphere, the lithosphere, the hydrosphere, the biosphere, and how they interrelate within the hydrologic cycle.

Ground Water—The water beneath the surface of the earth and its importance to man and the environment as a source of fresh water; water tables, rock porosity, aquifiers, and precipitation.

Moisture in the Atmosphere—The relationships between air temperature, heat, humidity, clouds, rain, and fog.

Surface Water—Lakes, streams, rivers, and how they are created; surface runoff, soil erosion, and the impact of human habitation and farming on the environment.

You must make a backup diskette before running each program. Once you do this, you are ready to begin the study session. Each program is broken down into 4 sections: the Tutorial, Human Impact, the True/False Quiz, and the Matching Quiz. The Tutorial and Human Impact sections are divided into subsections, each dealing with a particular aspect of the subject. You can select any section by pressing the appropriate key. If, while running the program, you decide that you need to review a section again to better understand a particular detail, you can easily move between screens or restart the section.

Question & Answer

Each interactive tutorial poses several types of questions designed not only to determine whether you know the correct answer, but also to lead the discussion forward. For example, the program may ask you "thought questions," which you must reason out, even if the program has not yet covered all the pertinent information. Review questions test how well you understand the different concepts that the program addresses. Other questions may involve analysis of diagrams and charts or may require you to perform a simple calculation.

A typical section consists of text accompanied by visuals and animation. In the *Ground Water* tutorial, for example, you first receive a definition of ground water, and then you study a series of animated visuals demonstrating how ground water forms, what happens to it underground, what we use it for, and how we extract it from soil.

The second section, Human Impact, discusses what effect man and civilization have on each element of the environment. In the *Hydrologic Cycle*, for example, you learn how a city the size of Los Angeles meets its considerable fresh water needs, and what this high demand for water can do to the environment surrounding the city.

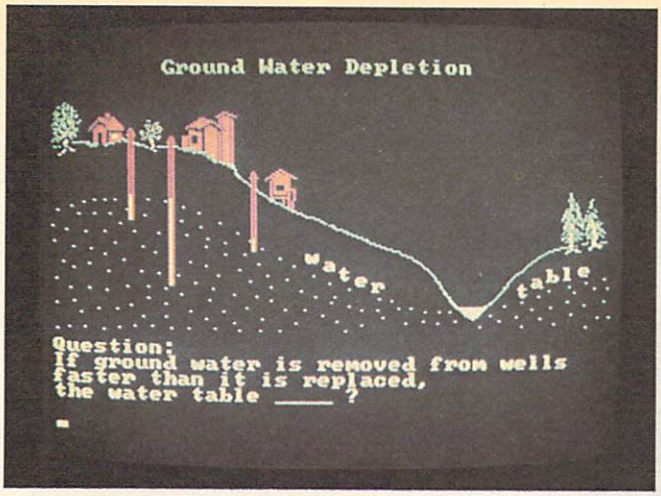
When you complete both sections, you can test yourself with either the Matching Quiz or the True/False Quiz. Both require you to answer 10 questions from the information in the tutorials.

The Overall Effect

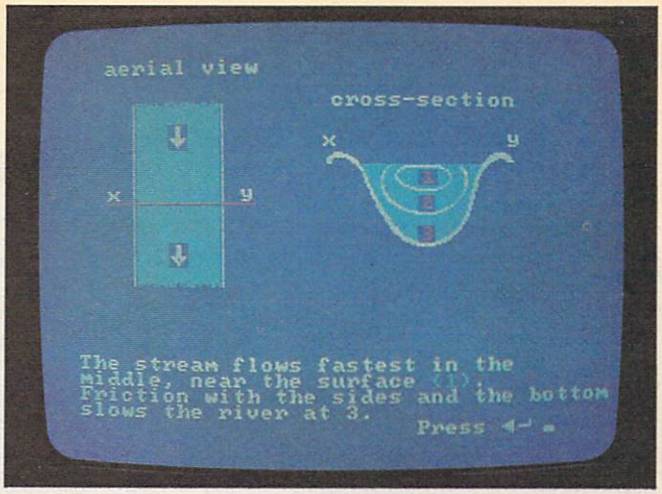
Depending on how quickly you study, it should take 30 minutes to an hour for you to work through each of the *Earth Science Series* packages.

Documentation is quite good—the manual is concise and provides a set of supplemental questions as well as a glossary of terms used in the course. Each *Earth Science* program follows exactly the same format, so once you've learned how to run one program, you can easily run them all.

The *Earth Science Series*' colorful and, in some cases, animated graphics are quite helpful in introducing the



This screen from Ground Water's Human Impact Tutorial shows how numerous water-pumping wells can make the ground water level drop.



This screen from the Surface Water Tutorial depicts the program asking a question about stream channel characteristics.

"Can this software actually help you learn a subject more effectively than you otherwise would using a textbook? In the case of the Earth Science Series, I think that the answer is positive."

information. The screens change quickly with no undue delays in drawing the graphics. Each visual is accompanied by explanatory text and usually a question. When you type in your answer, the program provides immediate feedback—a detailed explanation of the correct answer—regardless of whether your answer was correct or incorrect.

Sound effects are almost non-existent, and probably not necessary—although they would add to the enjoyment of the program.

Teaching With Computers

Inevitably, packages like the *Earth Sciences Series* bring up questions about the role of computers in education. For any educational program, the obvious question is: Can this software actually help you learn a subject more effectively than you otherwise would using a textbook? In the case of the *Earth Science Series*, I think that the answer is positive. Using these interactive programs, you can study at your own pace, go back over sections you did not understand, and test yourself. In addition, the subject has been culled down to just the essentials you need to know—nothing is left to distract you.

I think the important thing to remember here, is that *Earth Science Series* and similar educational programs are mainly supplements to normal, human interactive education. There is much more to learning than just memorizing a few terms and taking a quiz.

Effective Study Aid

When used as a study aid, the *Earth Science Series* probably does have a place in the classroom. A teacher or science department could combine the programs with a regular course of study, or keep them in a resource room or library, where students could review them for class assignments.

The *Earth Science Series* is an excellent set of programs that should make any science buff's wish list,

Name:	Earth Science Series: Hydrologic Cycle, Ground Water, Moisture in the Atmosphere, and Surface Water
Program Type:	Educational
Machines:	IBM PC, PCjr
Distributor:	IBM Corp. 900 King Street Ryebrook, NY 10573
Price:	$49 each
System Requirements:	128K, color/graphics monitor adapter

	Poor	Fair	Good	Excellent
Engrossment:				
Ease of Use:				
Documentation:				
Cost/Benefit:				

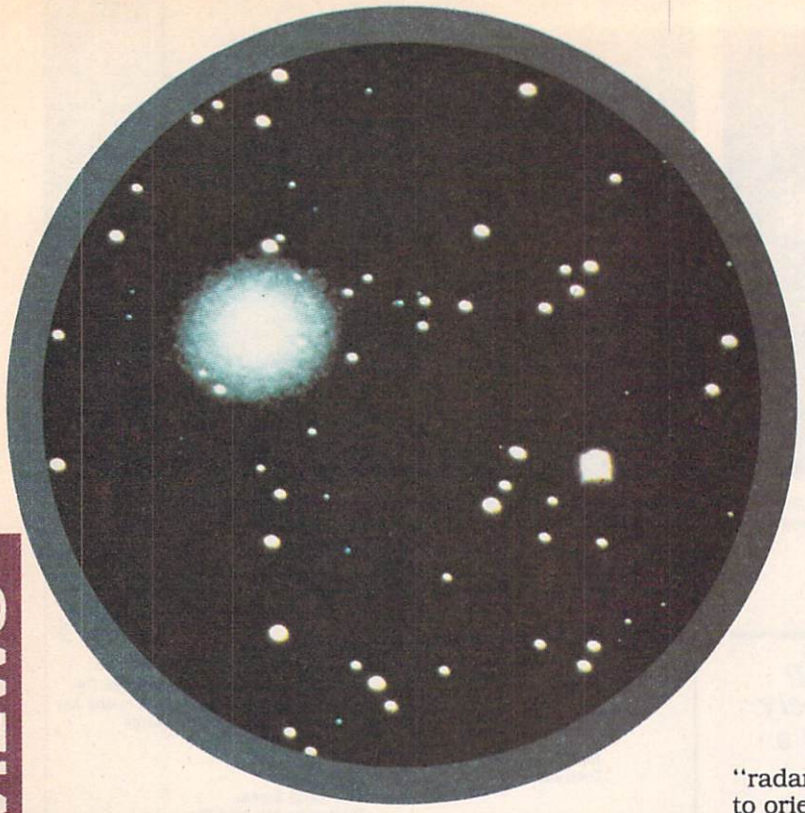
but its high price and limited subject matter will most likely keep it out of the home education market. A price of $49 per title is a pretty heavy investment for a program that you can exhaust in a day or two. Nevertheless, as a study aid, these programs can help students learn about Earth's environment, while doubling as a friendly and useful method for bringing neophytes into the world of computers.

☞ COUNTERPOINT ☜

Call me old-fashioned, but I still think the best way to learn is simply by reading books. Granted, the *Earth Science Series* employs colorful animation to present some very useful information, but it's only a fraction of the information comprised in most textbooks. After all, you can finish each program in only 30 minutes—how much knowledge can you cram into a half hour? Anyone "high school level and above" will surely stretch the entire series of programs to its limits within a couple of days. I think most users will soon ask themselves the same question I asked myself after using the series: "What can I do with these programs now that I've learned everything they can teach me?" I've found a use for them—they're just the right size to fill up the empty spaces in my book shelves at home. But, with the series selling at nearly fifty bucks per inch, Britannica is cheaper.

—Michael Hosmar

HCM



REACHING FOR THE STARS

A Review of 4 Astronomy Programs
by Scott Darroch

If the star-filled sky already grips your fascination, these astronomical software packages may hold the key to understanding what's really out there.

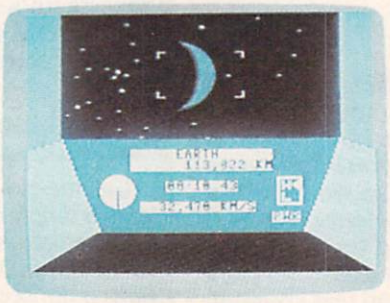
From our earliest beginnings, mankind has viewed the sky with fear, reverence, and speculation. Our predecessors regarded comets and eclipses as indications of disaster or divine wrath, while imagining the stars to be populated with fantastic beasts and mythical heroes. More careful study of these phenomena led to the development of astronomy—probably the first science. From the Stone Age to the Computer Age, astronomy has continuously evolved, and so has the the way in which we teach it. From lying in a field on a cold, clear night, to reclining in comfortable planetarium seating, we have refined our methods of viewing the changing sky. Now we have "astronomical" software providing interactive resources for both education and recreation. Here we examine a few such programs: *The Halley Project* for the C-64 and Atari; *Halley's Comet* and *Astronomy: Stars for All Seasons* for the Apple II family; and *Sky Traveler* for the C-64.

Learning Through Gaming

The Halley Project is an educational game—with the emphasis firmly on *game*. The game is in the "Mission Impossible" style, featuring sealed classified documents and a "mission briefing" on audio tape. Your assignment is to train for a secret project by flying a series of missions to planets and moons in the solar system, landing on each, and returning to the project base—Halley's Comet. When you complete all of the training missions, the program issues you a special I.D. code which, when registered with Mindscape Inc., gains you entrance to another game: *Halley Project—the Ultimate Adventure*. The missions are timed, with a list of suggested completion times to test yourself against.

The Halley Project disregards most real-world considerations of space flight, such as fuel consumption, acceleration and deceleration times, orbital and gravitational requirements, and, above all, the light-speed barrier. The flight system is fairly simple, comprising a "radar" screen to locate the planets, a viewing screen to orient the ship toward the proper zodiac constellation, and a drive to propel and steer the ship. The program accompanies game play with sound effects, some of which are fairly annoying, but necessary because they provide some important signals.

On the first mission, for example, you blast off from your base (Halley's Comet, of course), slow to a stop, and use the radarscope to determine your target's direction and distance. Then, orienting your ship by the constellations, you accelerate toward your destination. During acceleration, you are treated to a screeching, beeping chorus of noises until you reach the speed of light—then bells ring, lights flash, and hundreds of millions of kilometers slide beneath your bow. As you near your target, you must slip out of "hyperspace" and maneuver your craft around the planet until you locate a landing area. A flashing light and bell tone signal the landing beacon, but you can easily overlook these signals during a tense orbital session.



DESTINATION: EARTH.
Your approach to Earth brings you closer to completing the Halley Project.

Overall, we found this to be an enjoyable game that teaches a few basic astronomical concepts and presents enough challenge to prevent the premature onset of boredom. Although this program is more of a challenge to arcade gamers than it is a serious teaching tool, it does require you to recognize the zodiacal constellations and to think about relative distances and planetary positions.

Teaching By Tutorial

Two of the astronomy tutorial programs, *Astronomy: Stars for All Seasons*, and *Halley's Comet*, are intended for the classroom. Both make effective use of the computer's graphics abilities, but fall far short of the mark as either course material or reference sources. In terms of the texts, which are concise and understandable, both programs are accessible to students at fifth-grade level or above. Unfortunately, for young minds hungry for knowledge, the *content* of these programs provides only a thin soup—meager and unsatisfying.

NAME	MACHINES	DISTRIBUTOR	PRICE	SYSTEM REQUIREMENTS	POOR FAIR GOOD EXCELLENT
The Halley Project	Commodore 64, Atari 800 & compatible	Mindscape, Inc. 344 Dundee Rd. Northbrook, IL 60062	$39.95	Disk drive, joy stick	P: E: D: CB:
Halley's Comet	Apple II family	C.C. Publications, Inc. P.O. Box 23699 Tigard, OR 97223	$39.95	Disk drive	P: E: D: CB:
Astronomy: Stars for All Seasons	Apple II family	Educational Activities, Inc. P.O. Box 392 Freeport, NY 11520	$59.95	Disk drive	P: E: D: CB:
Sky Travel	Commodore 64	Commodore Electronics, Ltd. 1200 Wilson Dr. West Chester, PA 19380	$49.95	Disk drive, printer optional	P: E: D: CB:

KEY: P = Performance E = Engrossment D = Documentation CB = Cost/Benefit

Most disappointing is *Astronomy* from Educational Activities Inc., a fancy package concealing a nearly empty box. The program documentation looks impressive if you don't go beyond the contents page. Terms like "program objectives" and "teaching strategies" are liberally scattered throughout. Nevertheless, the total documentation extends to the scanty sum of 10 pages—including operating instructions and 5 pages illustrating the program's screens. If the text fails to convince you that the computer is not necessary to present the course material, a brief glance at the program should. After providing a few short and somewhat inaccurate explanations of basic astronomical terms and phenomena, *Astronomy* attempts to illustrate them with an extremely limited planetarium set-up. This device gives you a view of the night sky for any given *month* at any location from 0 to 90 degrees north latitude. Such niceties as longitude, year, day, and time of day are all neglected, so we must view the accuracy of the display with some suspicion. The planetarium is so constrained, it would be quite difficult to use as a teaching aid.

Astronomy's lack of depth and adaptability, combined with the almost nonexistent text and high price ($59.95) easily qualify it as the most worthless of the 4 programs we review here.

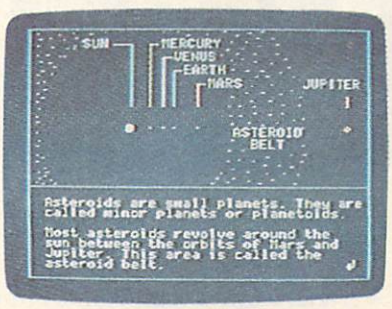
Grabbing the Comet's Tail

Halley's Comet and the Universe of Meteors, Asteroids, and Lesser Comets from C.C. Publications is far more comprehensive than *Astronomy* and comes closer to executing its objectives and making full use of the computer's capabilities. In addition to the 4 topics referred to in the title, this program comes with a workbook, 5 charts for plotting orbits, and a course on telescopes.

The program presents its tutorials in an illustrated lecture format, making good use of the Apple's graphic capabilities. The graphics are accurate and clear, illustrating each principle and condition with pleasing, easy-to-follow animation. Each tutorial includes several question and answer sessions. Correct answers result in an encouraging message to continue, whereas incorrect responses prompt the computer to present the material relevant to that question, then return you to the question. A preview option allows teachers to review the program's contents without having to answer every question before proceeding through the course.

The charts, workbook, and disk program together form a course that a student could easily pursue independently without missing any of the package's educational value. Topics include orbital shapes, locations and periods, structure and evolution of comets, and the origin of meteors. The workbook is quite useful, mostly because its projects and questions complement, rather than duplicate, the material contained on the program disk. Charts come in a three-ring binder allowing easy reproduction for classroom use (where permission is given). This course's smooth flow of instruction produces a coherent, accurate discussion of all the points that it covers.

The program's major short-coming lies not in content or construction, but rather in *overspecialization*. As a course in astronomy, the program is a three-legged stool with only one leg in place. One missing leg is information about the historical development of astronomy—the other is the illustration of planetary relationships and conditions. The publishers could upgrade this program to a comprehensive basic course in astronomy by including information on planetary motions, constellations, and the relationship of planets and their satellites. We hope to see such an addition to this informative and easy-to-use course.



Hop aboard Halley's Comet to discover space and explore the world of comets, meteors, telescopes, and our system's asteroid belt.

Hitting The Heights

Of all the programs that we cover in this review, *Sky Travel—A Window to the Galaxy* from Commodore Educational Software is the most comprehensive and complete—by several orders of magnitude. *Sky Travel* is a basic planetarium that can provide you a view of the sky from any position on Earth, at any time of day or night, and for a period spanning 20,000 years. Although it displays only a portion of the sky at a time, rather than the entire dome, *Sky Travel* allows you to look below the horizon and through Earth itself. No portion of the celestial sphere is hidden from the observer.

Sky Travel is very user-friendly, possessing a multitude of useful, informative features. Operating instructions are blue-sky clear, and the program's *textbook* (not just a manual) covers both the principles and the history of astronomy. This textbook would form an excellent foundation for a middle or high school course in astronomy. It explores elements of celestial mapping, planetary motion, the seasons, precession of the Earth's axis, and other topics—following each discussion with instructions on setting up an example from the planetarium. It also evokes history in relation to such astronomical events as the appearance of Halley's

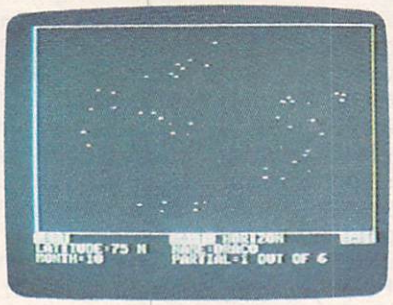
Comet shortly before William the Conqueror invaded England, the conjunction of planets setting the year 7 B.C. as the probable birth date of Jesus, and various historically significant solar and lunar eclipses.

Sky Travel provides numerous options that are simple to operate and quite useful for teaching, studying, or simply star gazing. A Set mode allows you to establish any observing date within 10,000 years forward or back from our present time. You can specify an observing time to the nearest second.

A Map mode allows you to shift your viewpoint to any position on Earth. This mode also displays a readout which locates your viewing position to the nearest second of latitude and longitude.

Sky and Chart modes show a view of the sky at the time and location you specify. The Sky mode displays a 72 degree arc—the average range of human peripheral vision. Chart mode projects the same view, but includes lines of latitude and longitude and reverses the field so that black stars rest on a white background. This reversal is especially useful for printing star maps from *Sky Travel*'s planetarium. With an Option function, you can add or delete such features as deep space objects, lines connecting stars in the constellations, symbols for planets, and the names of constellations.

A Find function automatically locates the moon, sun, planets, Halley's Comet, or any of 88 constellations, then brings the object to the center of field. You may use the cursor to look around—directly overhead or down through the Earth itself.



A screen from Astronomy: Stars for All Seasons depicting the night sky in October for latitude 75° in the Northern Hemisphere.

Words From Above

Sky Travel's data bank contains information on over 1200 objects and constellations. You can receive this information by placing the cursor over any object on the screen and pressing the Information button. Command keys allow you to narrow the field of view from 72 degrees to 36, 18, and 9 degrees. The smaller vision fields produce an effect much like a low-power telescope. As in telescope astronomy, you must keep the object you are viewing centered as you increase magnification. A variable clock can run the planetarium forward or back from a given time at rates varying from 1 :: 1 to 1 :: 64 (an hour in less than a minute). The accelerated mode allows rapid viewing of eclipses, conjunctions, and transits, while the slower rate keeps the display updated throughout an evening's star gazing.



To navigate the stars in Sky Travel, enter latitude, longitude, and time zone using the cursor keys or joystick.

The sky display is quite colorful and sharp, with the entire color scheme changing with the position of the sun. You can locate any object on the screen precisely by placing the cursor on the item and reading the Declination and Right Ascension or Elevation and Azimuth figures displayed in the data window on the side of the screen.

Although *Sky Travel* is an exceptionally complete display planetarium and data source, it suffers—like all things created by human beings—from a few drawbacks and omissions. For one thing, the program is based on a star chart of 2000 A.D. Because several of the closer stars have large proper motions, their positions do not appear accurately for time periods outside the span 1500 to 2500 A.D. These variations of position range from a few minutes to several degrees of arc. When you use the planetarium to view astronomical events that occurred before 1000 A.D., there may be time discrepancies of 0.5 to 2 hours. These are due to differences between universal time (related to Earth's rotation) and ephemeris time (based on an atomic clock). Variations in Earth's rotation rate, arising from such random sources as tidal drag, formation of ice caps, and continental drift, also preclude absolute accuracy.

Perhaps the major problem with *Sky Travel*'s fine system, however, is that it does not allow you to store several viewing screens for display. To change the planetarium's viewpoint you must switch from Sky mode to Map mode to set the location, then to Set mode to establish date and time. Although changing viewing screens takes only a minute or two, a preprogramed set of coordinates and times would be desirable in an instruction or lecture setting.

In a truly needless attempt to make this fascinating program more attractive to the young, Commodore has added sound and cursor options as well as a "needle in the haystack" type game. The cursor option causes the crosshair cursor to appear as an airplane in Map mode and a starship in the Sky mode. The sound feature adds sound effects to cursor motion. Totally undocumented, but reportedly buried somewhere within *Sky Travel*'s myriad images, are 11 "map tokens"—famous landmarks that you would see if looking at the proper portion of the horizon from a specific location at a specific time. Considering the amount of use these features are likely to see, the memory capacity might have been much better used to provide the save and recall feature mentioned above.

Despite these few complaints, this program packs in many benefits at a relatively small cost. Ease of use, instructive features, comprehensive data bank, and a sharp looking display make this a quality product for both class or home. It's too bad you can't take this little planetarium and project it on your ceiling—but, short of that, it's almost all you could ask for in an astronomy program.

As a classroom aid, home planetarium, or observational tool, *Sky Travel* truly lives up to its billing as a "window on the galaxy." For Commodore users, this is good news. Apple and Atari users may prefer to wait until the "stars are right" for their own truly celestial window.

[Watch for reviews of astronomy tutorial programs for other computers in a future issue—Ed.]

HCM Glossary terms: azimuth, conjunction, declination, precession, proper motion, right ascension, zodiac.

HCM

Speed Reading

A Review by Steve Nelson
HCM Staff

HCM Review



Name:	Speed Reading
Program Type:	Utility
Machines:	TI-99/4A
Distributor:	Navarone Industries, Inc. 510 Lawrence Expressway #180 Sunnyvale, CA 94086
Price:	$29.95
System Requirements:	32K memory expansion (Disk drive recommended)

	Poor	Fair	Good	Excellent
Performance:			■	
Ease of Use:				■
Documentation:	■			
Cost/Benefit:			■	

Speed reading—some people swear it works, others swear because it doesn't work for them. Either way, Navarone's reading course still needs more work.

The last time I checked, the jury was still out, trying to decide if speed reading courses really work. Some experts say they do, some say they don't. From my own experience with such courses, I think they can help, but I'm not sure how long the effects will last. It seems that once you attain a certain reading speed, you must continue to read at that speed on a regular basis. If you don't, your reading speed will slowly diminish to the level you started from. Be that as it may, it's up to you to decide if you want to spend $29.95 to find out if *Speed Reading* works for you.

Speed Reading by Navarone, is a cartridge-based reading course for the TI-99/4A. The program comes in two versions: one for adults and one for children. Except for differences in the skill level of the reading selections, both versions are identical.

Speed Reading is easy to use. The program simply highlights the portions of the reading material that you are to read. The highlighting moves along the text at the wpm (words per minute) pace you select. The screen defaults to a light blue, with the entire text faintly visible in the background, and the emphasized portion in white. You can adjust the screen colors to the combination that appears sharpest to you.

According to the manual, there are three basic elements that determine your reading speed: eye movement—the rate at which you move your eyes from one group of letters to another; eye span—the number of letters you take in with each fixation, or pause; and "perception"—the duration of each fixation. *Speed Reading* offers three exercises designed to enhance your abilites in these areas.

Exercise For The Eyes

The first exercise focuses on "eye fixation." This tricky little routine is supposed to help develop peripheral vision by flashing two sets of letters on each side of a dot in the center of the screen; first two letters, then four, and finally six—at three different speeds. You must stare at the dot, read the letters using your peripheral vision, and retain them in your memory long enough to duplicate them with the keyboard.

The second lesson exercises eye movement. After you enter your reading rate (in wpm), the computer displays the text faintly on the screen and highlights groups of words for you to read in a zigzag pattern, across and down the page. Each time the program highlights a new set of words, it directs the computer to sound a beep.

The third exercise is a column-reading drill. It is similar to the eye movement lesson in format; but instead of zigzagging, it displays the text down the center of the screen. You set the wpm rate and the width of the line of highlighted text (from 16 to 26 characters).

Speed Reading provides practice text, however you can load your own text file, as long as it is an ASCII file saved in display/variable 80 format, and the text does not exceed 100 lines.

To help you pace yourself as you read from a source other than the computer, the program supplies a reading pacer. You enter the speed you want to practice with, and the program beeps out a steady rhythm to help you maintain it.

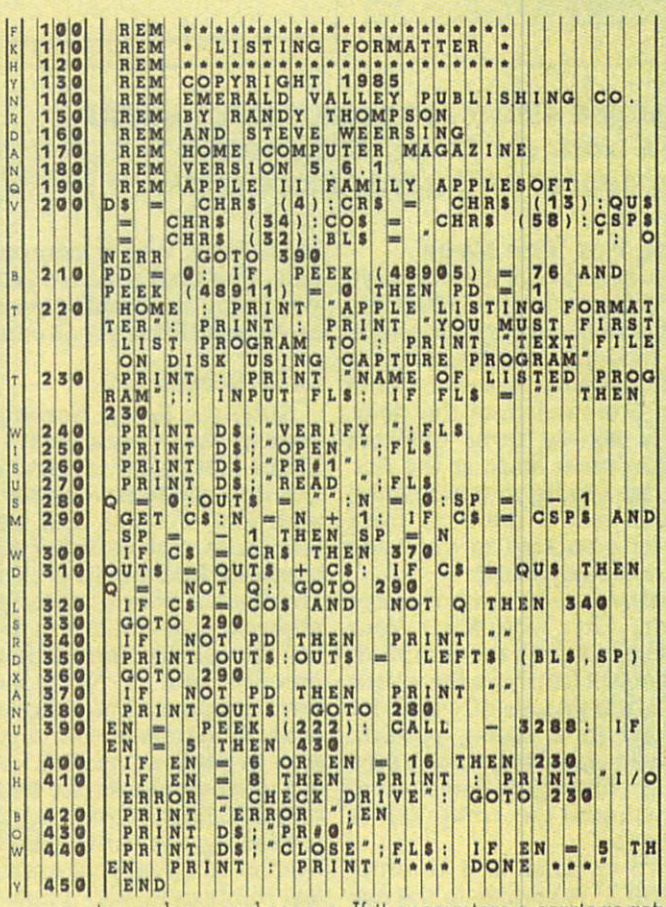
The manual provides practice sessions that incorporate the eye fixation and eye movement exercises. You work with these for 15 minutes each, then take a comprehension test. The final practice session has you using the reading pacer option as you read from a book or magazine.

Is It Worth It?

Although *Speed Reading* is very easy to use, it does have a few problems that I want to point out. First, the manual seems hastily written, with lots of typos and vague sentences. Its explanation of how people read and descriptions of exercises are ambiguous. For example, the eye fixation exercise limits you to 6 letters, but the manual implies that you can practice with 8. Navarone should have maximized its program to 8 or even 10 letters—I found it just too easy to read and retain 6 letters. The eye movement exercise limits you to only 100 lines of text, so you end up re-reading the same text over and over, or stopping your exercise every few minutes to load a new file. And finally, I was a bit put off by the general "it's all up to you" tone of the manual. People who really need to improve their reading speed require more guidance than the program's documentation provides.

What really bothers me about *Speed Reading*, however, is the apparent lack of thoroughness on the part of the developers. It left me feeling that the entire package was prepared in a somewhat perfunctory fashion. It could really stand some improvement, starting with a new manual. With a little more work, Navarone could possibly turn *Speed Reading* into a more useful and much friendlier program. But until this happens, you might want to spend your money on more conventional and perhaps even more thorough speed-reading courses. **HCM**

Applesoft Listing Formatter



Many BASIC programmers "crunch" their code, putting multiple statements on one line to use memory as efficiently as possible. Programs written in this format are compact and run quickly, but can be difficult to read. Here's a handy program in Applesoft BASIC that reformats listings to put each BASIC statement on its own line.

Listing Formatter requires a BASIC program listing to be resident on disk in the form of an ASCII text file. You can accomplish this by **LOAD**ing the program you want to list, keying in the *Capture* program listed on page 74, and entering **RUN**. The *Capture* program converts a BASIC program in memory into a text file on disk.

Once you have created the text file on disk, you can **LOAD** and **RUN** *Listing Formatter*. The program asks you for the name of the text file containing the listing, and then starts printing. Example output appears below.

After the program **OPEN**s the text file, the formatting algorithm reads one character at a time and checks for carriage returns, quotes, colons, and spaces. If it encounters a carriage return, it prints the current line and goes to the next one. It counts the number of characters up to and including the first space after the line number, and when it hits a colon that is not inside quotes, it goes to a new line. The program uses the number of characters it has counted to indent the current line so that it lines up properly under the line above it.

When the program reads past the end of the text file, Error Number 5 (END OF DATA) occurs. The program finds the error number by **PEEK**ing location 222, then branches to the cleanup routine and ends.

—Steve Weersing

← **Before formatting**

```
280 Q = 0:OUT$ = "":N = 0:SP =  - 1
290  GET C$:N = N + 1: IF C$ = CSP$ AND SP =  - 1 THEN SP = N
```

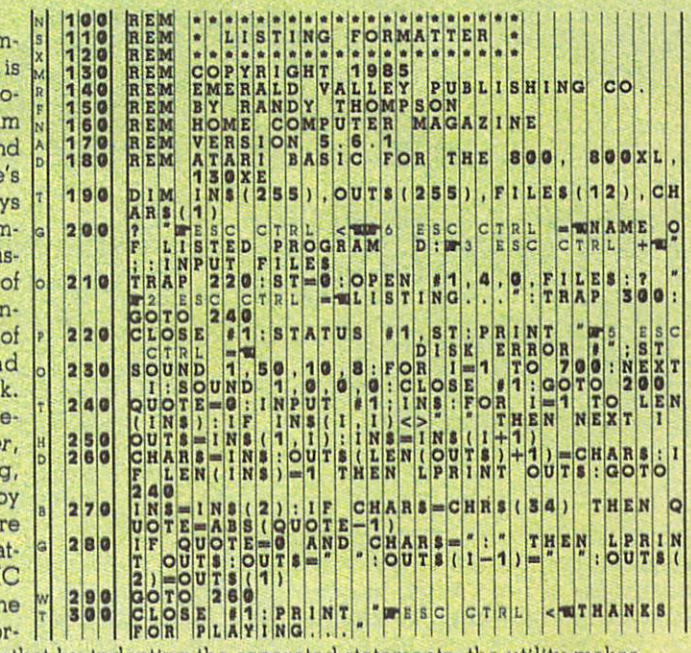
← **After formatting**

```
280 Q = 0:
    OUT$ = "":
    N = 0:
    SP =  - 1
290  GET C$:
    N = N + 1:
     IF C$ = CSP$ AND SP =  - 1 THEN SP = N
```

HCM Glossary terms: algorithm, ASCII, flag, text file.

Atari Listing Formatter

One of the best ways to improve your programming skills is to evaluate another person's program listing. Evaluating program listings is also the best way to find program bugs. Someone else's listings, however, aren't always easy to follow. When a programmer "crunches" a program by using multiple statement lines of nearly maximum length, untangling these long lines of BASIC commands and statements can be quite a task. The utility program that we present here, *Listing Formatter*, enables you to decipher long, compacted program lines by listing your programs in a more readable format. The utility neatly prints out each Atari BASIC statement onto a separate line (see the "before and after" formatting sample below). Notice that by indenting the separated statements, the utility makes the margin available for remarks.



To use this utility, you must first convert the program to be listed into an ASCII text file. Do this by loading the program into memory and **LIST**ing it to disk or tape. For example, to **LIST** a program entitled **GAME** to disk, first **LOAD** it into memory and then enter the following:

```
LIST "D:GAME.L"
```

The extension **.L** differentiates the **LIST**ed version of **GAME** from the program version. Now, **LOAD** and **RUN** *Listing Formatter*. When prompted, enter the program file name. For the above example, you would enter a file name of **D:GAME.L**. Before pressing [**RETURN**], make sure that your printer is hooked up and on-line. Once you press [**RETURN**], the utility sends the formatted listing to the printer.

The logic behind this utility is very simple. A loop reads in program lines while searching for colons, the statement separators. When it finds a colon, the formatter routine outputs the separated statement to the printer, then continues the search. Obviously, the program should ignore colons that occur inside quotes. To flag quotes, we use the appropriately named variable **QUOTE**. If **QUOTE** equals zero, the routine searches for colons. If **QUOTE** does not equal zero, the routine ignores colons.

—Randy Thompson

Before formatting →

```
230 SOUND 1,50,10,8:FOR I=1 TO 700:NEXT I:SOUND 1,0,0,0:CLOSE #1:GOTO 200
240 QUOTE=0:INPUT #1;IN$:FOR I=1 TO LEN(IN$):IF IN$(I,I)<>" " THEN NEXT I
```

After formatting →

```
230 SOUND 1,50,10,8:
    FOR I=1 TO 700:
    NEXT I:
    SOUND 1,0,0,0:
    CLOSE #1:
    GOTO 200
240 QUOTE=0:
    INPUT #1;IN$:
    FOR I=1 TO LEN(IN$):
    IF IN$(I,I)<>" " THEN NEXT I
```

HCM Glossary terms: flag, statement separator.

ATARI ATRIUM

Frequency Blender

by Steve Weersing
HCM Staff

Humble sine waves—the simplest of tones—combine under your control to produce complex waveforms.

Note: *Frequency Blender* first appeared in the "IBM-pressions" column of Vol. 5, No. 3, and in the *Apple Seedlings* column of Vol. 5, No. 5. For our Atari reader, we now offer a version here in "Atari Atrium"—Ed.

We are all familiar with the differences in tonal color among musical instruments. The liquid smoothness of the French horn and the sizzling bite of the electric guitar are distinctive because they produce differently shaped sound waves. The simplest sound has the shape of a sine wave and is a pure tone.

Every sound, no matter how simple or complex, can be broken down into one or more component sine waves. Complex waves—waves that consist of two or more sine waves—contain a fundamental sine wave and at least one harmonic. The fundamental component determines the pitch that we perceive in a complex wave form. The harmonic components govern the tone color.

Frequency Blender displays 4 sine waves with frequencies and amplitudes that you supply, then shows you their algebraic sum, known as the resultant. All 5 waves have the same height on the screen display so that you can see each wave clearly. When calculating the resultant, however, the program does take into account the amplitude of each component.

Making Waves

When you RUN the program, it asks you to enter frequencies and amplitudes for the 4 components and to specify the number of cycles of the lowest frequency that you wish to see on the screen. This last value sets the horizontal scale for the display. Negative amplitudes produce waves that are 180 degrees out of phase (upside down).

Because the horizontal resolution of the screen is limited, there is a restriction to the number of cycles that the screen can display clearly. If you try to display more than 50 cycles of the highest frequency, the display may distort whole cycles in the sampling and show a wave with a frequency lower than it actually has.

Examples 3 and 4 show approximations of square and triangle waves. You can obtain other interesting results with the values in these two examples:

Example 1:

```
FREQUENCY #1: 100
AMPLITUDE: 100

FREQUENCY #2: 110
AMPLITUDE: 100

FREQUENCY #3: 120
AMPLITUDE: 100

FREQUENCY #4: 130
AMPLITUDE: 100

Number of cycles of
lowest frequency: 50
```

Example 2:

```
FREQUENCY #1: 100
AMPLITUDE: 100

FREQUENCY #2: 200
AMPLITUDE: 100

FREQUENCY #3: 300
AMPLITUDE: 100

FREQUENCY #4: 400
AMPLITUDE: 100

Number of cycles of
lowest frequency: 10
```

How it Works

The seemingly complex task of adding these waveforms to produce their algebraic sum is actually straightforward. The program holds the frequency and amplitude values for the 4 component waves in the arrays FR( ) and AMP( ). After the 4 component waves are plotted, each of the 320 elements of a third array, SUM( ), will determine the position of the composite wave.

Before plotting any waves, the program sets all 320 elements of SUM( ) to zero. It then goes through a FOR-NEXT loop 4 times, drawing one of the component waves each time. An internal FOR-NEXT loop controls the plotting of the 320 horizontal positions for each component wave, calculating the vertical displacement of each point. The program also figures the relative effect that the wave has upon the composite wave, and then adds the value into the corresponding element of the SUM( ) array. Thus, after the program draws the 4 component waves, SUM( ) already contains the values needed to plot the composite waveform.

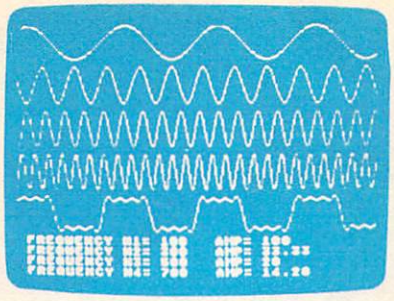
Example 3: *These frequency and amplitude values result in a close approximatation of a square wave:*

FREQUENCY #1: 100
AMPLITUDE 1: 100

FREQUENCY #2: 300
AMPLITUDE 2: 33

FREQUENCY #3: 500
AMPLITUDE 3: 20

FREQUENCY #4: 700
AMPLITUDE 4: 14.28



Number of cycles of lowest frequency: 4

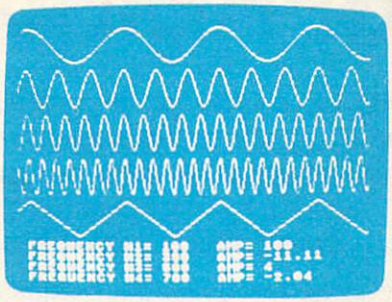
Example 4: *These frequency and amplitude values result in a close approximation of a triangle wave:*

FREQUENCY # 1: 100
AMPLITUDE 1: 100

FREQUENCY # 2: 300
AMPLITUDE 2: -11.11

FREQUENCY # 3: 500
AMPLITUDE 3: 4

FREQUENCY # 4: 700
AMPLITUDE 4: -2.04



Number of cycles of lowest frequency: 4

In order to scale the composite waveform, the program examines SUM( ) and looks at each element to determine SB (SumBig), the peak amplitude of the resultant.

Notice that the program uses the PLOT statement to position only the first point of each plot (line 310). To improve the clarity of the display, the program uses the DRAWTO statement for subsequent points, drawing lines between points instead of plotting only the points.

As the program plots each component wave, it displays the frequency and amplitude on the bottom of the screen. Once the display is complete, the component and composite waves remain on the screen until you press any key. Then the program gives you the option to continue with another set of waveforms or to exit the program.

HCM Glossary terms: amplitude, component, cycle, frequency, fundamental, harmonics, overtones, resultant, sine wave, waveform.

KEY VARIABLES

FR( ) = Frequencies of component waves; **AMP( )** = Amplitudes of component waves; **FM** = Lowest component frequency; **CY** = Number of cycles of lowest frequency to display; **D** = Function value (displacement) of each wave at each horizontal plot position; **SUM( )** = Function values for resultant; **X** = Horizontal plot position and loop counter; **Y** = Vertical plot position; **SB** = Amplitude of resultant; **I** and **T** are loop-counter variables.

```
100 REM  ********************
110 REM  *  FREQUENCY  BLENDER  *
120 REM  ********************
130 REM  COPYRIGHT 1985
140 REM  EMERALD VALLEY PUBLISHING CO.
150 REM  BY WILLIAM K. BALTHROP,
160 REM  ROGER WOOD,
170 REM  AND STEVE WEERSING
180 REM  HOME COMPUTER MAGAZINE
190 REM  VERSION 5.6.1
200 REM  ATARI BASIC FOR THE 800, 800XL,
         130XE
210 CLR :GRAPHICS 0:PRINT "■ESC CTRL <■
    ":PRINT "FREQUENCY BLENDER":DIM FR(
    4),AMP(4),SUM(320):PRINT
220 FOR I=1 TO 4
230 TRAP 230:PRINT "ENTER FREQUENCY #";
    I;:INPUT A:FR(I)=A
240 TRAP 240:PRINT "ENTER AMPLITUDE";:I
    NPUT A:AMP(I)=A:PRINT
250 IF FR(I)>0 AND FM=0 THEN FM=FR(I):G
    OTO 270
260 IF FR(I)>0 AND FR(I)<FM THEN FM=FR(
    I)
270 NEXT I:PRINT
280 TRAP 280:PRINT "HOW MANY CYCLES OF
    THE LOWEST FREQUENCY DO YOU WISH
    TO SEE DISPLAYED";:INPUT CY:TRAP
    32000
290 FOR X=0 TO 319:SUM(X)=0:NEXT X
300 GRAPHICS 8:COLOR 1:POKE 752,1:PRINT
    "■ESC CTRL <■":DEG :Y=15:SB=0:FOR
    I=1 TO 4
310 PRINT :PRINT "FREQUENCY #";I;"=";F
    R(I);"  AMP=";AMP(I);:PLOT 0,Y
320 FOR X=0 TO 319:D=SGN(AMP(I))*SIN(1.
    25*CY*X*FR(I)/FM):DRAWTO X,(Y-13*D
    ):SUM(X)=SUM(X)+D*ABS(AMP(I)):NEXT
    X
330 Y=Y+32:NEXT I:FOR X=0 TO 319:IF ABS
    (SUM(X))>ABS(SB) THEN SB=ABS(SUM(X)
    )
340 NEXT X:PLOT 0,Y:FOR X=0 TO 319:DRAW
    TO X,(Y-13*SUM(X)/(SB+(SB=0))):NEXT
    X:POKE 752,0
350 OPEN #1,4,0,"K:":GET #1,A:CLOSE #1:
    PRINT :PRINT :PRINT "AMPLITUDE OF R
    ESULTANT IS ";INT(10000*SB+0.5)/100
360 PRINT :PRINT "  CONTINUE (Y/N)?";:
    OPEN #1,4,0,"K:":GET #1,A:CLOSE #1:
    IF A=89 OR A=121 THEN 210
370 IF A=78 OR A=110 THEN GRAPHICS 0:EN
    D
380 GOTO 360
390 PRINT :PRINT :PRINT :PRINT "ERROR:
    VALUE OUT OF RANGE":FOR T=1 TO 25:S
    OUND 1,135,10,12:NEXT T
400 SOUND 1,135,3,12:FOR T=1 TO 1300:NE
    XT T:GOTO 210
```

HCM

LISTING ANNOTATIONS

Frequency Blender
(Atari 800, 800XL, 130XE)

Line Nos.	
100-200	Program header
210	Initialize screen display and variables
220-280	Get input values, set FM
290	Initialize SUM( )
300-320	Calculate and plot D, accumulate into SUM( )
330	Determine SB
340	Plot resultant, wait for keystroke
350-380	Display SB, get keystroke, branch to beginning or end
390-400	Error routine for bad numeric values

Thinking of Subscribing?
Remember these time-worn truths:



"A watched pot never boils."

"Patience is a virtue."

"Good things come to those who wait."

"Allow 6-8 weeks for delivery of your first issue."

HOME COMPUTER™ magazine



Commodore Hornblower

BASID

BASID takes the pain (and the POKEs) out of programming sound on the C-64 by adding 18 new commands and 17 functions to your computer's vocabulary.

by Randy Thompson
HCM Staff

In previous issues, we have used the pages of this column to give you a taste of the SID chip's various capabilities. We've shown you waveforms, envelopes, filters, ring modulation, and even synchronization. Now, we think it's about time you tried programming some of your own custom sounds. To make the task of creating sounds on the Commodore 64 easier, we've developed a special "sound" language. BASID adds 18 sound commands and 17 functions to your Commodore 64.

To use these new commands, you must first type in or LOAD BASID's BASIC loader. When you RUN the BASIC loader, BASID automatically activates, making all of the new sound commands available. You can use any of BASID's statements in both immediate mode and program mode.

See the Command Chart on the following page for a complete list of BASID's commands. The commands are in uppercase boldface, and the command parameters are in lowercase italics. For comparison, we have also included the BASIC equivalent of each BASID command. Note that, for most of these commands, it is impossible to give a true BASIC equivalent. For example, it is impossible from BASIC to simply set the attack of a voice without setting the decay at the same time, yet it is easy when you use BASID. In these cases, we have provided the closest BASIC equivalents.

The Functions

It is normally impossible to read the values stored in the SID chip's registers. BASID is unique, however—it allows you to read these values by using BASID commands (except for INIT) as functions. For example: X = FRQ(1) returns (gives you) the current frequency of voice 1 and places it in the variable X.

On the Commodore 64, you must follow all functions with an argument (the number a function operates on to produce its results) placed in parentheses. BASID's functions are no exception. In the example above, the argument specifies the voice (1, 2, or 3) that we wish to receive information about. In fact, in all but 6 of BASID's functions, the argument designates the voice. These functions—VOL (volume), LO (lowpass filter), BAN (bandpass filter), HI (highpass filter), CUT (cutoff frequency), and RES (resonance)—do not refer to any one voice, so they do not use the voice as an argument. Each of these functions requires a dummy argument. In other words, it does not matter what number you use as an argument—the function returns the correct value regardless of what argument was used.

Here are some examples demonstrating the use of BASID functions: X = SOUND (2) returns the status of voice 2. If voice 2 is on, X equals 1. If voice 2 is off, X equals zero. The function X = VOL (7) returns the current volume setting. The argument 7 is a dummy argument and has no effect on the value placed in X. The statement POKE 53280, VOL (3) changes the screen's border to the color corresponding to the current volume setting. Another interesting piece of coding sets the frequency of voice 2 to the current pulse width of voice 1: FRQ PULSE (1), 2

A Sample Program

On the next page, we have provided a sample program. We call it *BEEHZZOW* because of the peculiar sound it produces. Example 1 is the BASID version of *BEEHZZOW*, Example 2 is the BASIC version. As you can see, the BASIC version is both longer and less readable. You will find that the BASIC version is also much slower. In fact, the BASIC version is so slow that the sound effect is almost lost. Remember, in order to RUN a program that uses BASID commands, you must RUN BASID's BASIC loader first.

It's Your Turn

We've provided the language; it's up to you provide the talent. Try creating some simple sound effects, or writing your own arrangement of Beethoven's 5th. How about using BASID to write a program that converts part of the Commodore keyboard into a musical keyboard? Whatever you do, use BASID: You'll find that it's a lot easier than POKEing sound out of the Commodore. And, when you've written your masterpiece, send it to us—you might see your name in print.

BASID Command Chart

General Commands

INIT
Clears (initializes) the sound chip.
BASIC equivalent: FOR I=54272 TO 54296 : POKE I,0 : NEXT I

VOL *volume*
Sets the volume. The *volume* must be between 0 and 15 or an ILLEGAL QUANTITY error will result. A value of 0 is the lowest volume setting, while 15 is the highest.
BASIC equivalent: POKE 54296,*volume*

SOUND *on/off,voice*
Gates a voice on or off. Use an *on/off* value of 1 to turn the gate on (start the note), or a 0 to turn the gate off (end the note off). The voice must be a number between 1 and 3.
BASIC equivalent: POKE 54269+*voice**7,*on/off*+*waveform**16

Pitch And Waveform

FRQ *frequency,voice*
Sets the frequency (pitch) of a voice. The *frequency* can range between 0 and 65535.
BASIC equivalent: HB=INT(*frequency*/256) : LB=*frequency*–HB*256 : POKE 54265+*voice**7,LB : POKE 54266+*voice**7,HB

WAVE *waveform,voice*
Sets the waveform of a voice. The *waveform* parameter must be 1-4. Here's a list of the possible waveform settings:

1 - triangle
2 - sawtooth
3 - pulse
4 - noise

BASIC equivalent: POKE 54269+*voice**7,16*2↑(*waveform*–1)

PULSE *pulse width,voice*
Sets the pulse width. The *pulse width* value must be between 0 and 4095. This command is only effective when used in conjunction with a pulse waveform.
BASIC equivalent: HB = INT(*pulsewidth*/256) : LB=*pulsewidth* – HB * 256 : POKE 54267+*voice**7,LB : POKE 54268+*voice**7,HB

Envelope

ATTACK *attack,voice*
Sets the attack of a voice. The *attack* parameter must be 0 -15.
BASIC equivalent: POKE 54270+*voice**7,*attack*

DECAY *decay,voice*
Sets the decay. The *decay* parameter must be 0-15.
BASIC equivalent: POKE 54270+*voice**7,*decay**16

SUS *sustain,voice*
Sets the sustain. The *sustain* value must be 0-15.
BASIC equivalent: POKE 54271+*voice**7,*sustain*

REL *release,voice*
Sets a voice's release. The *release* parameter must be 0-15.
BASIC equivalent: POKE 54271+*voice**7,*release**16

Filtering

FILTER *on/off,voice*
Turns on or off the filters of a given voice. A value of 1 filters the voice, while 0 turns off the filter.
BASIC equivalent: POKE 54295,2↑(*voice* – 1)

LO *on/off*
Turns the lowpass filter on (1) or off (0).
BASIC equivalent: POKE 54296,16**on/off*

BAN *on/off*
Turns the bandpass filter on (1) or off (0).
BASIC equivalent: POKE 54296,32**on/off*

HI *on/off*
Turns the highpass filter on (1) or off (0).
BASIC equivalent: POKE 54296,64**on/off*

CUT *cutoff*
Sets the cutoff frequency of the filters. The *cutoff* value must range between 0 and 2047.
BASIC equivalent: HB=INT(*cutoff*/8) : LB=*cutoff*–HB*8 : POKE 54293,LB : POKE 54294,HB

RES *resonance*
Sets the filters resonance. The *resonance* value must be between 0 and 15.
BASIC equivalent: POKE 54295,*resonance**16

Special Effects

RING *on/off,voice*
Turns ring modulation on (1) or off (0).
BASIC equivalent: POKE 54269+*voice**7,*on/off* *4

SYNC *on/off,voice*
Turns synchronization on (1) or off (0).
BASIC equivalent: POKE 54269+*voice**7,*on/off* *2

BASID Quick Reference Card

Command (sets value)	Function (returns value)
INIT	-none-
VOL (0-15)	VOL (D)
FRQ (0-65535),V	FRQ (V)
WAVE (1-4),V	WAVE (V)
PULSE (0-4095),V	PULSE (V)
ATTACK (0-15),V	ATTACK (V)
DECAY (0-15),V	DECAY (V)
SUS (0-15),V	SUS (V)
REL (0-15),V	REL (V)
SOUND (0/1),V	SOUND (V)
FILTER (0/1),V	FILTER (V)
LO (0/1)	LO (D)
BAN 0/1)	BAN (D)
HI (0/1)	HI (D)
CUT (0-2047)	CUT (D)
RES (0-15)	RES (D)
RING (0/1),V	RING (V)
SYNC (0/1),V	SYNC (V)

V denotes Voice and must be represented by a number between 1 and 3.
D denotes Dummy argument and can be represented by any number.

Send your BASID programs to:

Home Computer Magazine
Attn: Commodore Hornblower
1500 Valley River Drive, Suite 250
Eugene, OR 97401

HCM Glossary terms: argument, attack, decay, dummy argument, envelope, frequency, function, parameter, pulse width, release, ring modulation, SID, sustain, synchronization, waveform.

For your type-in listings, see HCM PROGRAM LISTINGS CONTENTS.

HCM

Example 1

```
10 INIT:VOL 15:WAVE 3,1:SUS 15,1
20 FILTER 1,1:BAN 1:RES 15:SYNC 1,1:SO
   UND 1,1
30 FOR I=900 TO 0 STEP -15:FRQ 1000+I,
   3
40 CUT I:PULSE I,1:FRQ 4000-I,1:NEXT:I
   NIT
```

Example 2

```
10 S=54272:FORI=STOS+24:POKEI,0:NEXT:P
   OKES+6,240:POKES+24,47:POKES+23,241
20 POKES+4,35:FORI=900TO0STEP-15:POKES
   +14,(1000+I)AND255:POKES+15,(1000+I
   )/256
30 POKES+21,IAND7:POKES+22,I/8:POKES+2
   ,IAND255:POKES+3,I/256
40 POKES,(4000-I)AND255:POKES+1,(4000-
   I)/256:NEXT:FORI=STOS+24:POKEI,0:NE
   XT
```

KEY VARIABLES

D = Variable that reads machine language data; **I** = Memory location loop counter that stores machine language; **C** = Checksum variable that ensures correct entry of data statements.

LISTING ANNOTATIONS

BASID's BASIC Loader (C-64)

Line Nos.	
100-180	Program header
190-200	Read and store machine language data
210	Verify checksum
220	Initialize BASID
230-1860	Machine language data

TUTORIALS

C64 Alarm Clock

When you are working with computers, it is not difficult to lose track of time. With this easy-to-use program, and a little help from Commodore's built-in Time Of Day (TOD) clock, you can rely on your computer—not yourself—to remember when it's time to do something else.

Like a normal alarm clock, this program allows you to set both the time and the alarm. When the alarm goes off, the computer interrupts its operation, and the screen's border seems to come alive. To turn the alarm off, press the back-arrow key located in the upper-left corner of the keyboard. The border then stops strobing, and the computer resumes its operations as usual. Sorry, but there is no "snooze" button on this alarm.

When you **RUN** the program, a prompt asks you to enter both the current time and the alarm time. Enter these times in HHMMSS (Hours, Minutes, Seconds) format. For example, if it is 2:35 and 10 seconds, enter 023510. When prompted, press the **A** key for AM or the **P** key for PM. Once you have set the alarm, you can erase the *Alarm Clock* program and use the computer for almost any purpose, without disabling the alarm.

When you employ this program, you are actually using Commodore's TOD clock. This is not to be confused with Commodore's "jiffy clock," which is controlled through variables **TI** and **TI$**. Although the "jiffy clock" is easier to use and understand, it is less dependable (it stops running during certain I/O operations), and it has no alarm function. The TOD clock, on the other hand, is accurate—even during I/O—and has a user-settable alarm that causes an interrupt request at alarm time.

Memory locations 56328-56331 ($DC08-$DC0B) are the TOD clock's registers. The time is represented as follows:

Memory Location	Purpose
56328 ($DC08)	Tenths of seconds
56329 ($DC09)	Seconds
56330 ($DC0A)	Minutes
56331 ($DC0B)	Hours

Time is stored in these registers in Binary Coded Decimal (BCD). In BCD, a byte is divided into two nibbles, each nibble representing one decimal digit. For example, at 12 o'clock, the hours register (memory location 56331) contains a binary 00010010. Dividing this byte into nibbles, we get 0001 and 0010, representing the decimal digits 1 and 2 for 12 o'clock. All of the TOD clock registers work in this manner. The high bit of the hours register indicates either AM or PM: If it is PM, this bit is set (contains a one); if it is AM, the bit is clear (contains a zero).

These registers have a unique latching feature. When you read (**PEEK**) the hours register, all of the other registers freeze up until you read the tenths-of-seconds register. This prevents you from receiving an inaccurate reading. For example, if you read the hours register at 11:59 and the time changes to 12:00 just before you read the minutes, the latching feature prevents you from thinking that it is 11:00 instead of 11:59 or 12:00. Although the registers freeze, the internal clock does not. So, the moment you read the tenths-of-seconds register, all the registers update to the proper time.

By **POKE**ing these registers you can accomplish one of two tasks—either set the clock or set the alarm. To specify which one, you must manipulate the high bit of memory location 56335. If you set the high bit, then **POKE**ing the TOD registers sets the clock. To set the alarm, clear the high bit. Refer to lines 200 and 220 of *Alarm Clock* to see how to manipulate this bit.

The subroutine located in lines 270-360 sets both the clock time and alarm time. Lines 280-290 receive and verify your input. Line 300 converts this input into a numeric array of single digits. Lines 340-350 use this numeric array to **POKE** the time into the TOD registers in BCD format. The **AP** variable signifies AM or PM.

Now that you have set the alarm time, you must turn the alarm on. To enable the alarm, use the following code:

POKE 56333,PEEK(56333) OR 132

This code is in line 240 of *Alarm Clock*. After enabling the alarm, the program **POKE**s this location with a zero to ensure that no other system "alarms" are activated.

When the alarm goes off, it generates an interrupt request. An interrupt request normally forces the computer to execute the interrupt routine located at 59953 ($EA31). In order for the *Alarm Clock* program to know that an interrupt has occurred, we use a short machine-language routine to intercept the normal interrupt routine. This machine-language routine shows up in lines 420-750. The first part of this routine changes the interrupt vector located at 788-789 ($314-$315) so that it points to our new IRQ routine at 49165 ($C00D). It's the new IRQ routine's job to find the cause of any interrupt. If the TOD alarm is not the cause, then the routine returns control to the normal interrupt routine. If the TOD alarm is the culprit, then the border flashes, and all computer operation halts until you press the back-arrow.

Currently, the starting address for *Alarm Clock*'s machine language is 49152 ($C000). If you wish to alter this, simply change the variable **SA** in line 190 to equal another location. The cassette buffer, located at 828 ($33C), makes a good alternative.

—Randy Thompson

HCM Glossary terms: BCD, byte, high bit, interrupt request, machine language, nibble, TOD.

```
100 REM  ** ALARM CLOCK **
110 REM  ** COPYRIGHT 1985
120 REM  COMPUTE! PUBLISHING CO.
130 REM  BY RANDY THOMPSON
140 REM  HOME COMPUTER MAGAZINE
150 REM  VERSION 5.6.1
160 REM  C-64 BASIC
170 SA=49152:GOSUB370:REM SA=STARTING ADDRESS OF MACHINE LANGUAGE ROUTINE
180 POKE 56,PEEK(56334)AND127:REM GET TOP OF READY TO SET TIME
190 PRINT"{SHIFT CLR}INPUT THE TIME (HHMMSS)":GOSUB270
200 POKE 56333,PEEK(56333)OR128:REM GET READY TO SET ALARM
210 PRINT"{SHIFT CLR}INPUT ALARM TIME (HHMMSS)":GOSUB270
220 POKE 56333,PEEK(56333)OR128:POKE 56333,0:REM TURN ALARM ON
230 SYS SA:PRINT"{SHIFT CLR}ALARM SET":PRINT"THIS PROGRAM IS NO LONGER NECESSARY"
240 END
250 REM  INPUT AND SET THE TIME
260 PRINT"{HOME}"TAB(26);:INPUT TS:IF LEN(TS)<>6 OR VAL(LEFT$(TS,2))<1 THEN 280
270 IF VAL(TS)>129999 OR VAL(MID$(TS,3,2))>59 OR VAL(RIGHT$(TS,2))>59 THEN 280
280 FOR I=1 TO 6:T(I)=VAL(MID$(TS,I,1)):NEXT:PRINT"AM OR PM (A/P)? ";:GET KS:IF KS<>"A" AND KS<>"P" THEN 310
290 PRINT KS:AP=0:IF KS="P" THEN AP=128
300 POKE 56331,(T(1)*16+T(2))OR AP:POKE 56330,T(3)*16+T(4)
310 POKE 56329,T(5)*16+T(6):POKE 56328,0
320 RETURN
330 REM  STORE MACHINE LANGUAGE DATA
340 CK=0:FOR I=SA TO SA+61:READ D:POKE I,D:CK=CK+D:NEXT
350 IF CK<7341 THEN PRINT"ERROR IN DATA STATEMENTS":END
360 HB=INT((SA+13)/256):LB=SA+13-HB*256:POKE SA+2,LB:POKE SA+7,HB
370 RETURN
380 REM MACHINE CODE; LANGUAGE DATA LINE:
390 DATA 120
400 DATA 169,049,141,020,003
410 DATA 141,021,003
420 DATA 088
430 DATA 096
440 REM  ** IRQ - NEW INTERRUPT ROUTINE **
450 DATA 173,013,220
460 DATA 041,004
470 DATA 240,039
480 DATA 141,013,220
490 DATA 173,032,208
500 DATA 072
510 DATA 238,032,208
520 DATA 032,159,255
530 DATA 165,197
540 DATA 201,057
550 DATA 208,244
560 DATA 104
570 DATA 141,032,208
580 DATA 169,000
590 DATA 133,198
600 DATA 120
610 DATA 169,049,141,020,003
620 DATA 141,021,003
630 DATA 088
640 DATA 076,049,234
```

The Binary Tree

by William K. Balthrop
HCM Staff

See how recursion lets you imitate nature by growing a tree on your computer screen.

Take a look at the screen shot on this page. Judging by appearance, you might think that the tree it depicts was drawn with the help of a fancy graphics program. On the contrary, a simple BASIC program is responsible for the display. By loading and running this program, you can create a version of this tree on your monitor.

The Binary Tree draws the tree using "recursion." Recursion is a powerful programming technique whereby a procedure repeats itself until a specified condition is met. Most of us have seen this concept illustrated by paintings in which the artist has drawn the actual picture, which contains the picture, which contains the picture, and so on. Similarly, this program creates a graphic tree by repeating a branch-drawing routine until the tree is complete.



Take a recursive algorithm, add a little color, and voilà—a tree!

Spiraling Branches

Trees provide a perfect example of recursion. The overall pattern of the tree is repeated in ever-smaller levels; first in the tree itself, then in the large groups of branches, in the smaller groups of branches, and finally, in each twig. Each repetition represents a "level of recursion." In this program, each level of recursion draws a branch segment.

The program begins by drawing the trunk and then creating a branch that is 14 pixels in length. The recursive loop then calls itself and draws another branch at a 90 degree angle to the right of the first branch. This branch, and each consecutive branch, is one pixel shorter than the branch that preceded it.

At each level, the program saves the following information: the X and Y coordinates of the branch junction (where a larger branch meets one or two smaller branches), the length of the branch, and the number of branches at the current level. When the recursive process reaches its limit—i.e., when it draws a branch that is only one pixel long—it uses this information to back out to the preceding junction.

The program draws only two branches at each junction. Therefore, if the junction has just one branch—the right-hand branch—the program draws another series of spiraling branches to the left. If two branches already spiral off the junction, the program moves back to the previous junction, and so on. The entire program is complete when all junctions have two branches leading from them. (Figure 1 illustrates the path of the recursive process.)

Random Branches

The recursive algorithm, as we have explained it so far, always produces the same predictable geometric pattern—resulting in the same predictable, symmetrical tree each time. Trees in nature, however, do not have the luxury of an IBM to keep their branches in order. So, to simulate the variety of nature, we have added some random factors.

For each branch, the program generates a random number between 0 and 15. If the number that the computer produces lies between 4 and 15 (which it does approximately 81 percent of the time), the next branch that the program draws is one pixel shorter than the previous branch. If the random-number generator picks 0, the next branch is one-half the length of the previous branch. A random number of 1 results in a branch with a length that is 70 percent of the length of the previous branch. A value of 2 makes the next branch's length 80 percent that of the previous branch.

Colorful Branches

The most attractive aspect of the tree is not the algorithm we use to draw it, but the colorful way that it resembles a real tree in full bloom. The program does this easily by monitoring the length of the branches.

The program starts out creating a branch that is 14 pixels long. At every junction, the program reduces the next branch's length by one pixel (unless the random factor reduces the length by more). If the length is greater than 4 pixels, the program goes to a routine that draws brown branches. This routine also checks the branch's length to determine how broad the branch should be drawn.

If the length of the branch is 4 pixels or less, the program goes to another routine. This routine draws a green branch if the length is one through 4 pixels; a red dot if the length of the branch reaches zero.

Experiment

If you are the experimental type, you may like to try changing a few parameters to alter the shape of the tree. In line 450 you can determine how symmetrical the outcome will be by changing the multipliers that shorten the branches (.5, .7, and .8). In the same line, you could modify the values compared to RN to alter the randomness factor. In line 450 you could change the 1 at the end of the line to 2, 3, 4, or possibly 5. This will affect the speed in which the tree reaches the smallest branch size. If you reduce this value too far, you may receive a subscript error as the maximum recursion level is 100. You could change the maximum level, however, by increasing the DIMensioned size of RE(,) in line 270.

HCM Glossary terms: call, level of recursion, recursion, random number generator.

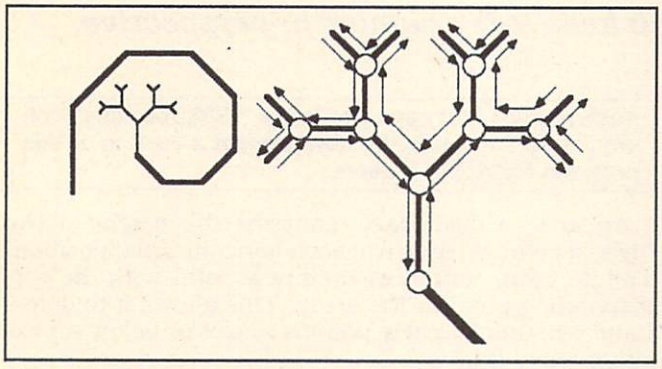


Figure 1: The recursive algorithm spirals inward until it has drawn the smallest possible branch. It then backs out, looking for junctions to branch out from, and continues the process.

LISTING ANNOTATIONS

Binary Tree (IBM PCjr, PC, Tandy 1000 PC)

Line Nos.

100-230	Program header
240-320	Initialize program variables
330-380	Main recursion loop
390-420	Adjust direction
430-470	Save the data of current level of recursion
480-520	Control color of branch
530-630	Draw brown branches
640-680	Draw green and red branches
690-720	Get the coordinates of the previous and current levels of recursion
730-760	Halt program
770-800	Data for the DIR array which determines direction
810-830	Wait for a keypress

KEY VARIABLES

A$ = Value of keypress; **RE(,)** = Recursion level information; **DIR(,)** = Direction values; **D** = Current direction; **LX,LY,RX,RY** = Offset used in drawing the branch; R = Current level of recursion; **RN** = Random number; **X1,Y1** = Coordinates of current branch junction; **X2,Y2** = Coordinates of previous branch junction; **Z** = Utility variable for loops.

```
100 '  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
110 '  *     B I N A R Y   T R E E     *
120 '  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
130 '     C O P Y R I G H T   1 9 8 5
140 '     E M E R A L D   V A L L E Y   P U B L I S H I N G   C O .
150 '     B Y   W I L L I A M   K .   B A L T H R O P
160 '     H O M E   C O M P U T E R   M A G A Z I N E
170 '     V E R S I O N   5 . 6 . 1
180 '     D O S   2 . 1   a n d   e i t h e r
190 '     I B M   P C j r   w i t h   C A R T R I D G E   B A S I C   o r
200 '     I B M   P C   w i t h   B A S I C A   a n d
210 '     C O L O R / G R A P H I C S   A D A P T E R   a n d
220 '     C O L O R   M O N I T O R   o r
230 '     T A N D Y   1 0 0 0   P C   w i t h   G W   B A S I C
240 '     I N I T I A L I Z E   P R O G R A M
250
260 DEFINT A-Z:RANDOMIZE TIMER
270 DIM RE(100,4),DIR(8,2)
280 FOR Z=1 TO 8:READ DIR(Z,1),DIR(Z,2):NEXT
290 SCREEN 1:COLOR 0,0:CLS
300 LINE (0,190)-(319,199),1,BF:DRAW "C3BM140,199R40HU3HU3HU3HU3HU3HU3HU3HU3HU40H10G10D40GD3GD3GD3GD3GD3GD3GD3GD3GBM160,140P3,3"
310 R=2:D=1:RE(1,1)=160:RE(1,2)=120:RE(1,3)=14:RE(1,4)=1
320 '
330 '  M A I N   R E C U R S I O N   L O O P
340 '
350 IF R=0 THEN 750 ELSE IF RE(R,4)=0 THEN D=D+1:GOSUB 410:GOSUB 450:GOSUB 500:IF RE(R,3)>0 THEN R=R+1:GOTO 350 ELSE D=D-1:GOSUB 410:GOTO 350
360 IF RE(R,4)=1 THEN D=D-1:GOSUB 410:GOSUB 500:IF RE(R,3)>0 THEN R=R+1:GOTO 350 ELSE D=D+1:GOSUB 410:GOTO 350
370 '
380 '  A D J U S T   T H E   D I R E C T I O N   B A C K   W I T H I N   L I M I T S
390 '
400 IF D=0 THEN D=8:RETURN ELSE IF D=-1 THEN D=7:RETURN ELSE IF D=9 THEN D=1:RETURN ELSE RETURN
410 '
420 '  S A V E   C U R R E N T   P O S I T I O N   F O R   A   L A T E R   R E T U R N   D U R I N G   R E C U R S I O N
430 '
440 RE(R,1)=RE(R-1,1)+(RE(R-1,3)*DIR(D,1)):RE(R,2)=RE(R-1,2)+(RE(R-1,3)*DIR(D,2)):RN=INT(RND*15):IF RN<1 THEN RE(R,3)=RE(R-1,3)*.5 ELSE IF RN<2 THEN RE(R,3)=RE(R-1,3)*.7 ELSE IF RN<4 THEN RE(R,3)=RE(R-1,3)*.8 ELSE RE(R,3)=RE(R-1,3)-1
450 RETURN
460 '
470 '  D R A W   T H E   B R A N C H E S   O N   T H E   S C R E E N
480 '
490 RE(R,4)=RE(R,4)+1:IF RE(R,3)>4 THEN GOSUB 550 ELSE GOSUB 660
500 RETURN
510 '
520 '  D O   B R O W N   B R A N C H E S
530 '
540 GOSUB 710:ON D GOTO 560,570,580,590,560,570,580,590
550 LX=-1:LY=0:RX=+1:RY=0:GOTO 600
560 LX=-1:LY=-1:RX=+1:RY=+1:GOTO 600
570 LX=0:LY=-1:RX=0:RY=+1:GOTO 600
580 LX=+1:LY=-1:RX=-1:RY=+1
590 IF RE(R,3)>9 THEN LINE(X1+LX,Y1+LY)-(X2+LX,Y2+LY),3
600 IF RE(R,3)>7 THEN LINE(X1+RX,Y1+RY)-(X2+RX,Y2+RY),3
610 LINE(X1,Y1)-(X2,Y2),3:RETURN
620 '
630 '  D O   G R E E N   A N D   R E D   B R A N C H E S
640 '
650 IF RE(R,3)<1 THEN C=2 ELSE C=1
660 GOSUB 710:LINE(X1,Y1)-(X2,Y2),C:RETURN
670 '
680 '  G E T   C O O R D I N A T E S
690 '
700 X1=RE(R-1,1):Y1=RE(R-1,2):X2=RE(R,1):Y2=RE(R,2):RETURN
710 '
720 '  E N D   O F   P R O G R A M
730 '
740 GOSUB 830:END
750 '
760 '  D I R E C T I O N   D A T A
770 '
780 DATA 0,-1,1,-1,1,0,1,1,0,1,-1,1,-1,0,-1,1,-1
790 '
800 '  I N P U T   R O U T I N E
810 '
820 A$="":WHILE A$="":A$=INKEY$:WEND:RETURN
830
```

HCM



Apple Seedlings

3-Dimensional Graphic Surfaces Made Easy!

by Steve Weersing
HCM Staff

This intriguing technique will help you keep 3-D graphics in perspective.

TUTORIALS

Imagine yourself spending a pleasant afternoon tossing rocks into a still pond or lake and watching the ripples spreading from the point of impact. Wouldn't it be great to see this fascinating natural process in more detail?

We have all seen these ripples frozen in time by stop-action photography. Now we can see them on our home-computer screens as well. Spreading ripples are a good example of wave motion. Each ripple is a sine wave that gets weaker as it spreads out in all directions from the center.

The Apple II family's high-resolution (hi-res) graphics screen offers you an effective medium for depicting these ripples. *Ripples* draws a graph showing diminishing waves expanding outward from a central splash—in three dimensions! At your option, it also demonstrates other graphic possibilities by drawing the graph with differently shaped waves.

When you RUN the program, it displays a menu asking you to select ripple patterns based on sine waves, square waves, or triangle waves. When you enter a selection, the program clears the screen and draws the ripple pattern. It then waits until you press any key and asks if you wish to continue. If you do, it shows you the menu again. If you don't, it restores the ripple pattern on the screen and stops. You may clear the screen by holding down [CONTROL] and pressing [RESET].

A Little Background

The most difficult aspect of drawing in perspective on a computer screen is keeping track of which parts of the drawing are in the foreground and which are in the background. How is the computer to tell which parts of the drawing are "behind" other parts?

The solution involves drawing from the bottom of the screen (the foreground) to the top of the screen (the background) and suppressing parts of the background that are not supposed to "show through" parts of the foreground.

The program draws 29 wavy horizontal lines from left to right, moving up the screen after completing each one. It draws each line by going along an invisible horizontal baseline containing 280 plot positions. At each horizontal plot position, the program calculates a vertical offset that places the current point above, below, or on the baseline.

Note: *Ripples* first appeared in the "IBMpressions" column of Vol. 5, No. 2. We now present a version of this program for Apple II users.

An array called MAP( ) contains the height of the highest point yet drawn at each horizontal plot position. The program compares each new point with the corresponding point in the array. This allows it to determine whether the new point is above or below an existing wavy line.

If the new point dips below any previous line, the program does not draw the portion that is underneath that line. Instead, it retraces the contour of the existing line, and continues drawing the new line when it emerges from "behind" the existing one.

The program uses a similar technique whenever a line goes off the top or bottom of the screen. After calculating the vertical position of a point, the program checks to see if the point is on the screen. If it is not, the program draws to and along the edge of the screen until the next point is on the screen.

There is no problem with the left and right edges of the screen because the program automatically draws from the left edge to the right edge, stops, and starts again at the left edge.

How Big A Splash?

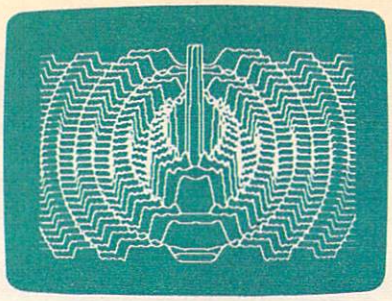
The vertical offset of each wavy line from its baseline at any given point depends on the distance from that point to the center of the screen. The center of the screen has the X and Y coordinates (139,95). Our old friend the distance formula, based on our ancient friend the Pythagorean Theorem, gives us the linear (straight-line) distance from any point on the screen to the center. This distance, D, is equal to SQR((139-X)^2+(95-Y)^2).

Plotting the cosine of D at each point along each baseline produces a ripple pattern of constant height. Such a graph is interesting, but it does not show the large central spike or the diminishing ripples found in nature, as when the stone hits the pond's surface.

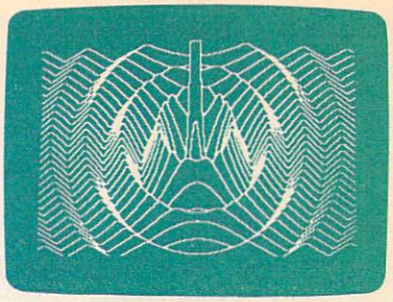
Because the size of the ripples diminishes as the distance from the center increases, the amplitude of an actual wave is inversely proportional to the distance from the center. Plotting COS(D)/D produces a graph with these qualities. Multiplying D by scaling factors in the



This ripple graph, based on a sine wave, shows the splash at the center going off the top of the screen.



This ripple graph is based on an approximation of a square wave, which is not a naturally occurring pattern.



This ripple graph uses a different set of sine wave components to approximate a triangle wave.

numerator and in the denominator allows you to adjust the size and proportions of the graph.

Controlling The Screen

When you use the HGRB statement, Applesoft BASIC selects the hi-res graphics screen with a 4-line text window at the bottom. By using POKE statements, you can conveniently switch this window on and off. In *Ripples*, this feature brings back the bottom of the ripple graph after displaying the text window with a "Continue?" prompt.

The HGRB statement not only sets the hi-res mode, but also clears the screen. On the other hand, POKE – 16302,08 turns off the text window, and POKE – 16301,08 turns it back on—without clearing the screen. When you execute one of these POKEs, both the text window and the bottom of the hi-res screen retain whatever you last wrote to them.

Because the TEXT statement does not clear the text screen, you can use it to bring back the text after using the graphics screen. You can clear the text screen with the HOME statement.

Color Me Wavy

If you have a color monitor, you may wish to experiment with foreground and background colors. If you change line 230 to read HGR : POKE – 16302,0 : HCOLOR = 3 : HPLOT 1,1 : CALL 62454 : HCOLOR = 6, the program will draw in blue on a white background.

HCM Glossary terms: horizontal plot position.

HCM

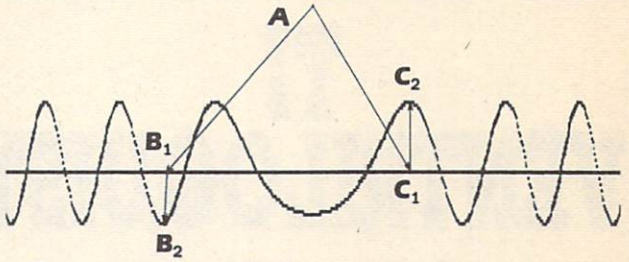
KEY VARIABLES

MAX( ) = Array to hold highest point plotted at each horizontal plot position; **A$** = Character input; **X** = Horizontal plot position and loop counter; **Y** = Vertical position of each baseline and loop counter; **D** = Distance to center of screen; **F** = Vertical offset of point from baseline; **P** = Actual Y coordinate of each point plotted.

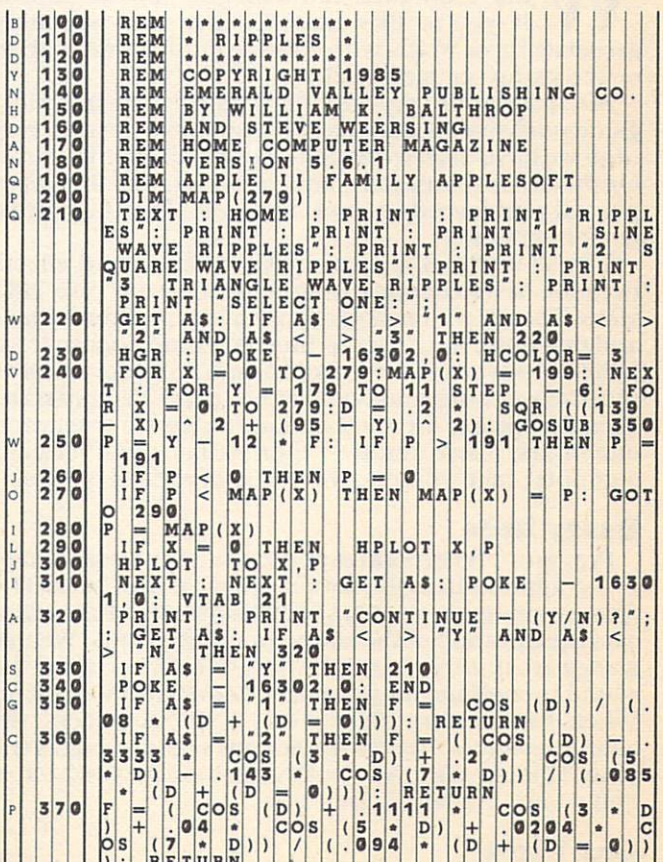
LISTING ANNOTATIONS

***Ripples* (Apple II Family)**

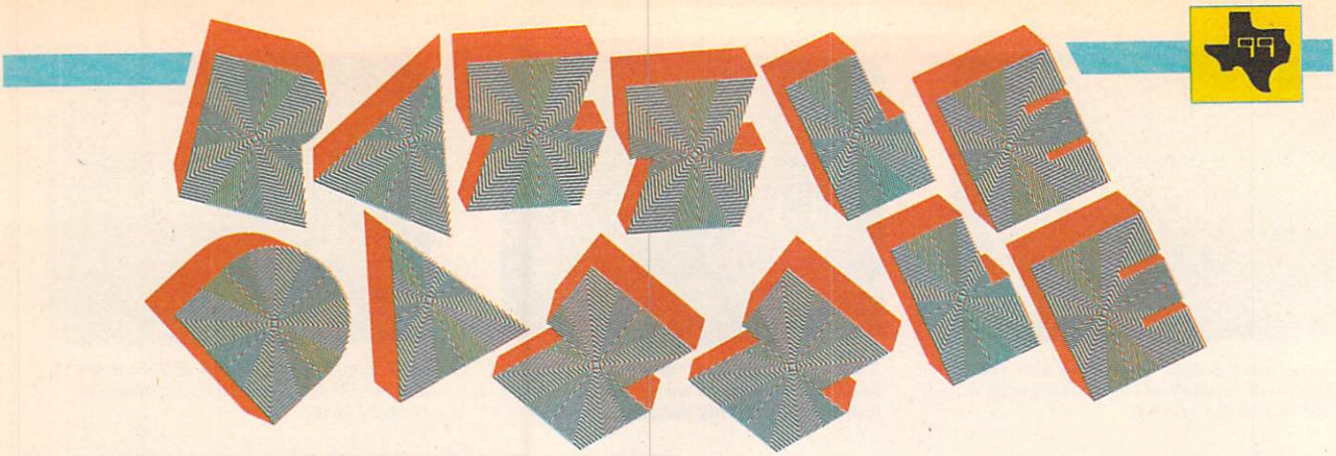
Line Nos.	
100-190	Program header
200	Dimension array
210-220	Menu
230	Set graphics screen
240	Calculate D
250-300	Plot each point
310-330	Wait for keystroke, get input, branch to beginning or end
340	Restore graphics at bottom of screen, end



This chart illustrates the algorithm that draws the screen. At each point on a horizontal baseline, the program determines the distance **D** to the point **A**. It then calculates a vertical offset from the baseline using **COS(D)/D**. At **B1**, a negative offset places the point to be plotted at **B2**. At **C1**, a positive offset places the point to be plotted at **C2**.



TUTORIALS



TI VENTRILOQUIST

by Scott Williams
HCM Staff

Tune-in to intriguing sound effects with your TI computer and a few dazzling routines . . .

What pops into your mind when someone mentions computer simulations or arcade games? If you're like most people, you probably associate these words with striking graphics animation or the game scenario itself.

But there is one very important feature built into most games and simulations that we tend to overlook, at least until we have to do without it—*sound effects*. Try turning down the volume on your television or monitor, then play all your favorite games. You won't find them nearly as interesting.

The TI-99/4A is a great machine for sound effects because of its complex 4-voice sound chip, the TMS 9919. This chip is capable of producing up to three tones and one noise at the same time. You can choose from 8 different types of noises and a wide range of tones as the example programs on these two pages demonstrate. These programs are short enough for you to enter each one in just a few minutes, then sit back and let your computer take you into the world of special effects.

Buzzy Bees

Program 1 incorporates one tone and the noise channel to approximate the sound of a swarm of bees. The first parameter in the CALL SOUND statement in line 200 sets the duration of the sound in milliseconds. We use a negative number to set the duration. This causes the sound to continue until the next CALL SOUND statement. When it encounters the next CALL SOUND statement, the computer simply leaves the sound on and changes the tone. This results in a smooth transition between tones and a very realistic honey bee.

Gun Fight

Program 2 simulates a gun shot repeated at random intervals. The noise channel plays a big part in this sound effect.

The TI computer responds to 8 noise parameters, each designating a different noise. To specify a noise, you need to use a negative value between −1 and −8 as a frequency parameter. Value −5 is the closest approximation to white noise—random frequencies dispersed evenly throughout the audio spectrum. Values −6 through −8 also produce noise—however, the lower frequencies contain more energy, which results in a lower pitched noise. Values −1 through −4 create what the TI BASIC Manual refers to as "periodic noise." Technically speaking, this periodic noise is not a noise at all. It more nearly resembles the sound of a narrow pulse wave. If you specify a value of −4 or −8, the frequency setting of the third tone in the CALL SOUND statement determines the pitch of the noise. This reference chart displays the relative pitches of the various types of noise available on the TI sound chip:

```
-1 or -5 = high pitch
-2 or -6 = medium pitch
-3 or -7 = low pitch
-4 or -8 = set by frequency of the third tone
```

Motor Rev

By mixing several sound effects together we can create a more complex simulation. Program 3 produces a very familiar sound: a car having difficulty starting up. The car motor turns over 8 times, then starts and slowly revs up. The engine holds a steady RPM for a few seconds then slows until it stops.

A FOR-NEXT loop in lines 110 through 140 handles the initial turning over of the engine. We use a noise setting of −8 with the third tone controlling the pitch of the noise. The loop counter then alters the frequency of the third tone, making the noise's pitch change. In the first CALL SOUND statement, the volume is at its maximum level of 0. In the next sound statement we decrease the volume to 5 and alter the frequency of the third tone to drop the pitch slightly. This results in the whirring sound of the crank case turning over.

The second FOR-NEXT loop starts the pitch of the noise channel at a very low level so that you can hear only clicks. These clicks increase in frequency with an increase in the frequency of the third tone, providing the thrumming noise of an idling engine.

The next loop holds the idle tone for a few seconds. The final FOR-NEXT loop takes the pitch of the noise from its high "idle" back to a slow clicking.

Hiccup

Program 4 incorporates two noise types and all three tones to produce the type of sound you might expect to hear if your computer had the hiccups. The first sound you hear is a high-pitched "hic." Line 110 generates this sound by setting the noise channel to −8 and setting the third tone at a frequency of 20000 Hertz (Hz). Voice two is used at a low volume to add a little tone color to the hic.

The second sound that you hear is a lower frequency. To achieve this effect, we specify the fixed-pitch noise –7 and mix it with three tones of 330, 337, and 380 Hz.

We have specified a positive duration to create a distinct break between notes. The sound lasts for the entire length of the duration, then ceases completely before the next note is played.

Siren

A police siren is a surprisingly complex sound to simulate. The sound, as first heard from a distance, is quiet. As the police car approaches, the siren becomes louder. Then the car passes, and the siren becomes quieter, fading into the distance. This variation in amplitude is fairly easy to reproduce on the computer. The complexity lies in a simultaneous oscillation in frequency.

To produce this sound effect in Program 5, we need to control two parameters: the frequency of the siren as the sound fluctuates up and down, and the volume of the siren as it approaches and recedes.

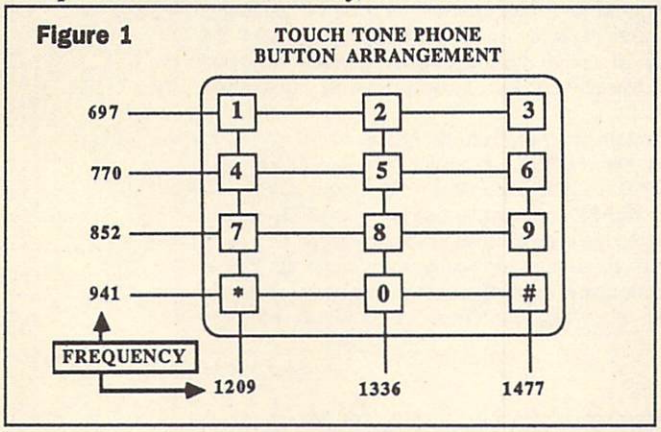
The simplest way to achieve this sound effect is to place the frequency oscillation aspect in a subroutine located in lines 250 through 280. The main program then cycles through two FOR-NEXT loops to control the increase and decrease in volume, calling the subroutine to play the siren.

In nature, few things are as perfectly linear as FOR-NEXT loops. So, for the sake of realism, we added a further dimension to the routine. We placed both the volume and pitch under the control of the SIN function. Unfortunately the SIN function is not very fast. We have therefore eliminated it from the most repetitive areas of the program, placing a list of pre-calculated SIN values in an array to save processing time later. We also have calculated the SIN of Z before calling the subroutine, rather than calculating it through every iteration of the Y loop.

Touch Tones

Have you ever wondered what would happen if you sent simulated touch tones to the mouthpiece of your phone? The answer: You would dial the number corresponding to the tones. The touch tones that dial your phone do not have to originate inside the phone; you can produce them externally with the same results.

Figure 1 TOUCH TONE PHONE BUTTON ARRANGEMENT



Touch tones are easy for your computer to generate. Each touch tone actually consists of a combination of two separate tones. One tone is determined by the row that the key lies in; the other is determined by the column (see Figure 1).

Program 6 generates random phone numbers, playing the accompanying tones. Do not hold your phone up to the speaker of your TV set while this program is running—you may end up with an expanded phone bill!

HCM Glossary terms: iteration, linear, noise channel, periodic noise, white noise. HCM

Program Listing 1: ***BUZZY BEES***

```
100 REM BUZZY BEES
110 I=INT(RND*10+1)
120 J=INT(RND*10+1)
130 FOR V=I TO J STEP -1*(J>I)+1*(I>=J)
140 GOSUB 180
150 NEXT V
160 I=J
170 GOTO 120
180 F=INT(RND*2)+115
190 SN=SGN(RND-.5)
200 CALL SOUND(-500,-3,10,F+(SN),V)
210 RETURN
```

Program Listing 2: ***GUN FIGHT***

```
100 REM GUN FIGHT
110 CALL SOUND(100,110,0,130,5,34000,30
    ,-8,0)
120 FOR Z=0 TO 30 STEP 15
130 CALL SOUND(-100,110,30,110,30,34000,
    30,-8,Z)
140 NEXT Z
150 FOR Z=1 TO INT(RND*5)*30
160 NEXT Z
170 GOTO 110
```

Program Listing 3: ***MOTOR REV***

```
100 REM MOTOR REV
110 FOR Z=1 TO 8
120 CALL SOUND(-100,110,30,110,30,2800+
    Z*50,30,-8,0)
130 CALL SOUND(-100,110,30,110,30,2700+
    Z*50,30,-8,5)
140 NEXT Z
150 FOR Z=3000 TO 30000 STEP 600
160 CALL SOUND(-100,110,10,120,10,Z,30,
    -8,0)
170 NEXT Z
180 FOR TD=1 TO 70
190 CALL SOUND(-100,110,10,120,10,Z,30,
    -8,0)
200 NEXT TD
210 FOR Z=30000 TO 2000 STEP -100
220 CALL SOUND(-100,110,10,120,10,Z,30,
    -8,0)
230 NEXT Z
```

Program Listing 4: ***HICCUP***

```
100 REM HICCUP
110 CALL SOUND(40,110,30,440,10,20000,3
    0,-8,0)
120 FOR T=1 TO 25
130 NEXT TD
140 CALL SOUND(30,330,0,337,0,380,10,-7
    ,0)
150 CALL SCREEN(1)
160 CALL SCREEN(16)
170 FOR TD=1 TO RND*500+300
180 NEXT TD
190 GOTO 110
```

Program Listing 5: ***SIREN***

```
100 REM SIREN
110 DIM S(40)
120 FOR X=0 TO 3.14 STEP .079
130 C=C+1
140 S(C)=SIN(X)
150 NEXT X
160 FOR Z=1.57 TO 0 STEP -.314
170 Z1=SIN(Z)
180 GOSUB 250
190 NEXT Z
200 FOR Z=.2 TO 1.57 STEP .5
210 Z1=SIN(Z)
220 GOSUB 250
230 NEXT Z
240 END
250 FOR Y=1 TO 40
260 CALL SOUND(-100,S(Y)*200+440,Z1*30)
270 NEXT Y
280 RETURN
```

Program Listing 6: ***TOUCH TONES***

```
100 REM TOUCH TONES
110 RANDOMIZE
120 READ T1(1),T1(2),T1(3),T1(4),T2(1),
    T2(2),T2(3)
130 DATA 697,770,852,941,1209,1336,1477
140 FOR Z=1 TO 3
150 GOSUB 330
160 PRINT C$;
170 CALL SOUND(200,T1(S1),0,T2(S2),0)
180 CALL SOUND(70,110,30)
190 NEXT Z
200 PRINT "-";
210 FOR TD=1 TO 60
220 NEXT TD
230 FOR Z=1 TO 4
240 GOSUB 330
250 PRINT C$;
260 CALL SOUND(200,T1(S1),0,T2(S2),0)
270 CALL SOUND(70,110,30)
280 NEXT Z
290 PRINT ::::
300 FOR TD=1 TO 300
310 NEXT TD
320 GOTO 140
330 S1=INT(RND*4)+1
340 S2=INT(RND*3)+1
350 IF (S1=4)*((S2=1)+(S2=3))THEN 330
360 C$=SEG$("123456789*0#",(S1-1)*3+S2,
    1)
370 RETURN
```

Debugging DOS Directory —Part 2—

Last issue, we introduced you to the DEBUG.COM utility, which comes on the supplementary programs disk of MS-DOS 2.10, and showed you how use it to manipulate the DOS directory. In this tech note, we continue with more tips on using DEBUG to alter the directory.

If you don't include a /V parameter when you FORMAT a disk, the disk is unlabeled. Usually, there is no way for you to add a volume name from DOS—but there is with DEBUG. As we stated last issue, when you write to a disk using DEBUG, you could make a mistake that destroys the directory of a disk, making the files on the disk virtually unrecoverable. Therefore, exercise caution when modifying a disk, and don't use your only copy of a treasured disk—use a backup.

Now, here's how you can add a volume name:

(1) Place a disk with DEBUG.COM on it in the default disk drive (drive A), then type **DEBUG** and press [**ENTER**]. You are presented with the DEBUG prompt—a hyphen (-).

(2) Next, place the disk to which you will add the volume name in drive A, then type **L 100 0 5 4** to load the directory into memory. This command loads **4** disk sectors of the directory into memory—which is sufficient for most disks. If your disk contains 64 files or more, however, the directory will require more than **4** disk sectors. If this is the case, type a **5** in place of the **4** above.

(3) Now, to view the directory (in groups of 4 entries) type **D** (for **D**ump) and press [**ENTER**]. Continue pressing **D** and [**ENTER**] until you see an entry like this:

```
08F1:0200   00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   .vvvvvvvvvvvvvvv
08F1:0210   F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   vvvvvvvvvvvvvvvv
```

Available places in the entry are marked by a location containing a zero (00) followed by a series of F6s. The first location in which this occurs (200 in the example above) is where you will add the volume name.

(4) Let's say you want to call the disk MYDISK. Following the example above, you would type: **E 200** and press [**ENTER**]. If the address on your disk is other than 200 (as is very likely), substitute the address where the zeros and F6s first appeared during the **D**ump. Now you are presented with this display:

```
08F1:0200   00.
```

The cursor flashes just to the right of the period after the zeros. Here, enter the hexadecimal number corresponding to the ASCII of the first letter in the new disk name (refer to the hexadecimal/ASCII chart below.) M is the first letter of MYDISK, so enter 4D (the hexadecimal for M). Next, press the Space Bar, and you will be taken to the next memory location that contains a F6. Here, enter a 59 for Y, then press the Space Bar again.

Continue in this manner until you have entered all the letters in MYDISK. The DOS directory allows an 11 character volume name. You must fill in the rest of the volume name with blanks (hexadecimal 20). In the twelfth character position, you must specify the type of file (the file attribute). Enter 08 —the attribute of a volume name. Now, press [**ENTER**], and your changes are recorded in memory.

(5) Next, make sure that the write-protect notch is uncovered on the disk, and type **W 100 0 5 4** to write the new directory with your volume name to the disk, then type **Q** to **Q**uit DEBUG. Note that you must **W**rite the same number of sectors that you originally **L**oaded. For example, if you **L**oaded with **L 100 0 5 5**, you should **W**rite with **W 100 0 5 5**

Now when you get a directory of the disk it will contain the new volume name.

The space character is not legal in a DOS file name. From BASIC, however, you can **SAVE** a file with a space in the name. You may then **SAVE** or **LOAD** a BASIC file using this illegal name, but you cannot delete the file from DOS using the **DEL** or **ERASE** commands, nor can you **KILL** the file from BASIC. By using DEBUG, however, you can make the file eraseable again. Follow steps 1 through 3 above to run DEBUG, load the directory that contains the illegal file name, and locate the illegal file name with the **D**ump command. Then, use the **E**nter command to place any legal file name character in place of the space (such as a 41 for a capital A). Next, follow step 5 to **W**rite the corrected directory back to the disk. Now you can delete the file.

—Roger Wood

HCM Glossary terms: ASCII, hexadecimal, sector.

CHARACTER	HEXADECIMAL	CHARACTER	HEXADECIMAL	CHARACTER	HEXADECIMAL	CHARACTER	HEXADECIMAL
A	41	N	4E	0	30	%	25
B	42	O	4F	1	31	&	26
C	43	P	50	2	32	'	27
D	44	Q	51	3	33	(	28
E	45	R	52	4	34	)	29
F	46	S	53	5	35	-	2D
G	47	T	54	6	36	/	2F
H	48	U	55	7	37	@	40
I	49	V	56	8	38	\	5C
J	4A	W	57	9	39	_	5F
K	4B	X	58	:	21	`	60
L	4C	Y	59	#	23	{	7B
M	4D	Z	5A	$	24	}	7D

Several Flags In One Variable

Flags are an essential programming element. For most programs, you can keep each flag in a separate numeric variable. But as memory gets tight (a common occurrence if you are using a TI-99/4A without memory expansion, and your programming task requires many different flags), you may wish for an alternative to this method.

Well, here is a trick in TI Extended BASIC that lets you keep several flag values in just one numeric variable! This method has two key elements: First, because a flag is binary in nature (yes or no, on or off, etc.), you must interpret the numeric variable as a binary quantity. The other important part of this trick is using the TI Extended BASIC logical operators **OR** and **AND** to set, clear, and test (read) the various flag "bits" in a variable.

Before you can use one of the logical operators in this manner, you must first know the value of each bit within a binary number. The first chart below shows these values.

Bit number:	7	6	5	4	3	2	1	0
Bit value:	128	64	32	16	8	4	2	1
Bit:	1	1	1	1	1	1	1	1

For example, the decimal number 32 is a binary 0010 0000, and 34 is a binary 0010 0010. Therefore in an 8-bit quantity you have 8 different flags—one for each bit.

Let's see what effect logical operators can have on binary numbers. Both the **AND** and **OR** operators compare two binary numbers and produce a numeric result.

To set a flag, use the **OR** operator. You can set a particular bit within a numeric variable by **OR**ing that variable with the value of the bit to be set. The **OR** operator checks to see if a bit in the first number (the flag byte) is set or the corresponding bit in the second number (or mask) is set. If so, the **OR** operator sets that bit in the resulting number. The second figure shows the result of the operation **40 OR 12**.

Let's use the variable **FL** as an example. To set bit number 2 in **FL** do the following: **FL = FL OR 4**. To set bit number 3 do this: **FL = FL OR 8**.

Decimal	Binary								
40	0	0	1	0	1	0	0	0	Flag byte
12	0	0	0	0	1	1	0	0	Mask
44	0	0	1	0	1	1	0	0	Result

To clear a flag, use the **AND** operator. You can clear a particular bit within a numeric variable by **AND**ing that variable with everything *but* the value of the bit to be set. The **AND** operator compares two numbers; if a bit in the flag byte is set *and* the corresponding bit in the mask is also set, then the **AND** operator sets that bit in the resulting number. An 8-bit number holds a maximum value of 255—all bits set. To clear just the zero bit, execute the following: **FL = FL AND 254** (255 minus the value of the bit to be cleared). To clear bit number 3, do this: **FL = FL AND 247**. The third figure shows the result of the operation **40 AND 247**.

Decimal	Binary								
40	0	0	1	0	1	0	0	0	Flag byte
247	1	1	1	1	0	1	1	1	Mask
32	0	0	1	0	0	0	0	0	Result

Now that you can clear and set bits, how do you read them? You test a bit by **AND**ing it with the value of the bit that you want to read. To test if the zero bit of a flag is set or clear, do the following: **RESULT = FL AND 1**. To test bit number 3, execute this: **RESULT = FL AND 8**. If **RESULT** ends up equal to zero, the flag is clear. If **RESULT** is a nonzero number, the flag is set.

With the **AND** and **OR** operators, you can manipulate each bit separately. With this capability, you can use each bit within a numeric variable as an independent flag. For example, you can use the zero bit in a variable to flag whether output should go to the screen (bit is clear) or the printer (bit is set). Bit number 3 can determine whether the program will use sound effects or not.

Decimal	Binary								
40	0	0	1	0	1	0	0	0	Flag byte
8	0	0	0	0	1	0	0	0	Mask
8	0	0	0	0	1	0	0	0	Result

— Scott Williams

HCM Glossary terms: bit, byte, flag, logical operator, mask.

Problems In

With this new column and HCM productivity software, you can build solid decision-making tools.

A CHALLENGE TO OUR READERS

In past issues, HCM has published a number of productivity programs as tools for making calculated decisions. *The Organizer* for example, helps you prioritize and systematize your thoughts. And *Snap-Calc* lets you adapt a simple spreadsheet for your own applications—for example, tracking investments, comparing purchase options, calculating payroll, etc.

"Problems in Productivity" is a column designed to help you find new ways of using these productivity tools. Each issue, this column presents one "problem in productivity"—a real-life situation that cries out for a computer solution. You can solve all the problems by using one of our previously-published programs as a tool.

Next issue, we will publish our own solution to the problem presented here, together with a *new* problem. You can then compare our solution to your own.

TUTORIALS

This issue's productivity problem calls for *Snap-Calc*, published in the Vol. 4, No. 3 (August, 1984) issue. Your challenge is to use this simplified computer spreadsheet to answer a question many families face every year: Where can we find the college education to best fit our financial situation? In solving the problem, you will learn more about spreadsheets in general, and *Snap-Calc* in particular. [At present, there is no version of *Snap-Calc* for the Atari 800, 800XL, or 130 XE. We will, however, make this version available soon—Ed.]

Asking Questions

How *do* you plan a college education? What framework can you use to compare all the financial alternatives offered by all the different schools you're likely to consider? In Figures 3, 4, and 5, we give you a typical (but imaginary) choice between three different institutions. Although this data varies between colleges, all items reflect common expenses that any college student encounters. Your "assignment," should you choose to accept it, is to develop a spreadsheet template for comparing the three sets of data and computing a "bottom line" that indicates the best financial alternative. Next issue, we will publish our solution: a completed template that manipulates the data in Figures 3, 4, and 5.

Snap-Calc In Short

What on Earth is a spreadsheet template? Put simply, it's a framework of formulas used to study relationships between numeric "elements"—both constant and variable. In the old style of bookkeeping (handwritten), these mathematical comparisons were often confusing, time-consuming, and tedious. Every time one number changed, the bookkeeper had to refigure all the calculations and make corrections.

But with *Snap-Calc*, you can design a pre-set template of formulas and let the computer do most of the work. By entering raw data into the template, you can do a day's worth of bookkeeping in a few seconds. If one of your figures changes, you merely input the new number, and the computer recalculates all the formulas and complex relationships for you. This makes *Snap-Calc* a powerful "what if" tool, allowing for rapid comparison of many different alternatives. You can use it time and again to test different options or hypothetical situations.

Here's an example: Let's say you want to calculate the cost of your dream vacation this summer. Before you leave, create a template with *Snap-Calc* that enables you to list all your daily expenditures. Include columns for meals, hotels, plane tickets, rental cars, fuel, tips, and tours. You can then enter a formula that sums up these daily costs (either actual or estimated). Then by incorporating a formula that subtracts your expenses from your available funds, you can balance each day's expenditures.

You may use this template either to plan a vacation that fits your budget—by comparing all the alternatives—or to keep track of your daily expenses while traveling. Figure 1 shows the information needed for such a spreadsheet, and Photo 1 shows what the spreadsheet would look like on-screen.

For actual *Snap-Calc* operating instructions, see Vol. 4, No. 3 (August, 1984) of *HCM*. If you don't have *Snap-Calc* with all its updates, see "DeBugs on Display" Vol.

Figure 1

```
LOGIC NAME IS TRIPCOST
TOTAL COLUMN IS 10
LAST COLUMN IS 9
1 IS DATE
2 IS CAR RENTAL
3 IS GAS COST
4 IS GALLONS
5 IS ODOMETER START
6 IS ODOMETER STOP
7 IS MILES
7 = 6 - 5
8 IS MILEAGE
8 = 7 / 4
9 IS COST/MILE
9 = 3 / 7
10 IS BREAKFAST
11 IS LUNCH
12 IS DINNER
13 IS SNACKS
14 IS ENTERTAINMENT
15 IS TRIPS
16 IS HOTEL ROOM
17 IS AIR FARE
18 IS TAXI FARE
19 IS DAILY TOTAL
19 = 2 + 3 + 10 + 11 + 12 + 13 + 14
   + 15 + 16 + 17 + 18
20 IS RUN TOTAL
20 IS 19 + LAG 20
```

Photo 1

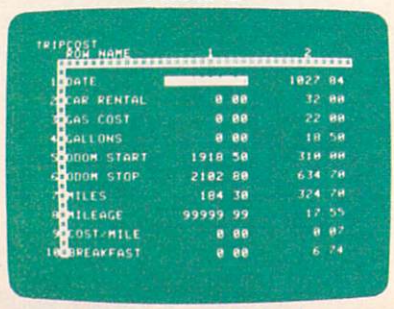


Figure 1 shows each "Tripcost" formula numbered by row. Photo 1 shows a typical spreadsheet with entries

4, No. 5; Vol. 5, No. 1; Vol. 5, No. 2; and Vol. 5, No. 3; or order the Vol. 4, No. 3 ON DISK/ON TAPE. (Note: *Snap-Calc* on the TI-99/4A requires memory expansion.)

Now For The Challenge

As mentioned, this issue's productivity challenge is to create a spreadsheet template that allows a comparison between different college education alternatives. Here are some suggestions that will help you formulate your template: You will need gross income rows for the student, both parents, and income from other sources. (Gross income figures are used to analyze the student's qualifications and need for some types of student financial aid.) You'll need 4 rows for expendable income. Make expense categories for tuition, housing and meals, books and supplies, transportation, entertainment, clothing, and vacations.

You will need formulas that will sum the various quantities and subtract the expenses from the total income. (As you develop the spreadsheet, you will encounter the need for other formulas to create subtotals, etc. If you need extra help, turn the page upside down and read "Hints.") If the *variance*—the difference between income and expenses—is a positive number, the student can afford the particular college used in the parameters. But, if total financial resources fall short of expenses (a negative variance), the student must qualify for financial aid to make up the difference. Try entering different levels of financial aid. If you project that scholarships or grants would cover less than the variance, add in available savings. And if this still doesn't cover the costs, examine other types of loans that allow low payments while the student is still in school.

Keep in mind all the options available. You may even discover some options that we have not suggested. The example in Figure 5 assumes that the student starts at a community college, attends part-time, lives at home, and is working for funds to transfer to another institution. But in this case, transfer procedures may require 5 years—instead of 4—to complete the student's education. Also, as shown in Figure 4, if the student wants to attend an out-of-state school, additional costs for transportation and out-of-state tuition affect the total financial picture.

Work first with the sample parameters and numbers provided in Figures 2, 3, 4, and 5. (These will appear in our published "solution" next issue.) Then, you might try using other items and statistics you've accumulated from researching real institutions that interest you. Changing figures and parameters—and then letting the program recalculate the results—will show you how flexible and useful a spreadsheet like *Snap-Calc* really is.

HCM

Figure 2
Sample Annual Income Base

1st Year Income:	
Father's Gross	20,000
Mother's Gross	16,000
Student's Gross	1,500
Other	1,000
Father's Expendable	1,500
Mother's Expendable	1,500
Student's Expendable	500
Other	500
Savings Available	500
Loan costs	7.5%

Note: Income increases 10% per year.

Figure 3
College A
(4 Years In State)

1st Year Expenses:	
Tuition	1432.50
Meals & Housing	1098.00
Books & Supplies	300.00
Misc. Personal	990.00

Note: Expenses increase 5% per year

Figure 4
College B
(4 Years Out of State)

1st Year Expenses:	
Tuition	4057.50
Meals & Housing	2340.00
Books & Supplies	300.00
Misc. Personal	990.00

Note: Expenses increase 5% per year

Figure 5
College C
(2 Years Community College Then Transfer)

1st Year Expenses:	
Tuition	594.00
Meals & Housing	400.00
Books & Supplies	300.00
Misc. Personal	800.00

Note: Expenses increase 5% per year. For expenses after the transfer to either an in-state or out-of-state university (for 3 remaining years), see Figures 3 and 4.

Hints

After familiarizing yourself with *Snap-Calc*, keep in mind these helpful hints: Use a large piece of paper to sketch in the positions of each formula and how they relate to each other. When you've finished the sketch, plug the formulas into *Snap-Calc* to make the spreadsheet template. Save it onto a disk or tape (Commodore and TI only) before loading it with data.

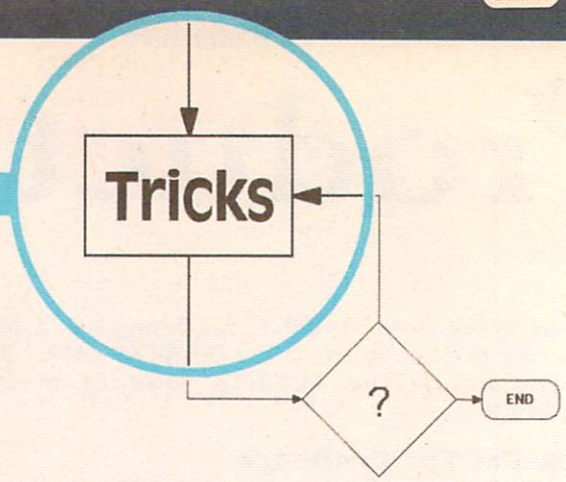
You will need to include formulas in the template to perform the following operations:

1. Total gross income
2. Add (10%) rise in income (Lag function)
3. Total expendable income
4. Add (10%) rise in income (Lag function)
5. Total expenses
6. Add (5%) inflation rate to expenses
7. Subtract total expenses from expendable income (VARIANCE 1)
8. Total financial aid
9. Add variance 1 to total aid (VARIANCE 2)
10. Add variance 1 to savings and conventional loan, then subtract your loan payment (VARIANCE 3)

NOTE: Your formula template may vary from this list depending on your machines capabilities.

TUTORIALS

Algorithm-A-Tricks



ASSEMBLING AN ALGORITHM

by the HCM Staff

WHAT IS AN ALGORITHM?

An algorithm is simply a procedure—one that a program uses to complete a task or solve a problem. Flow diagrams and flow charts are handy tools for representing the steps in this procedure. Any program can be viewed as a collection of separate procedures. In this column, we focus on and explain one unusual or interesting algorithm that is found in one of the programs we publish each issue.

From mnemonics to numbers: How does an assembler do it?

This issue's "Algorithm-A-Tricks" focuses on the *NanoAssembler*. Assemblers, by nature, require several complex algorithms, and the *NanoAssembler* is no exception. The main job of any assembler, however, is to identify mnemonics and translate them into their machine-language equivalent. Here we discuss how the *NanoAssembler* accomplishes this formidable task.

Using a series of DATA statements, the program READs all the possible mnemonics into a string array— OP$( ) for example. Each element of the array stores one mnemonic. (Note that because Atari BASIC does not support string arrays, we must simulate them. We do this by creating a single long string that contains all of the different mnemonics, and then use the Atari computer's string manipulation operator to extract the mnemonic we wish to check for. For the purposes of this column, we will continue to use arrays as our example, but Atari users should refer to line 1130 to see how the *NanoAssembler* simulates these arrays.)

The secret behind this technique is the order in which the program stores the mnemonics: according to their numeric machine-language equivalent (their op-code value). For example, the ADD mnemonic has an op-code value of 0, so OP$(0) = "ADD". Similarly, because the STA mnemonic has an op-code value of 3, OP$(3) = "STA". This conversion method applies to all 16 mnemonics.

Now, how do we use the OP$( ) array to convert a *NanoEditor* source code listing into machine language? Fairly easily. Using a loop, the program compares the mnemonic field of a line with each element of the OP$( ) array. The loop executes a maximum of 19 times—i.e., 16 times for the 16 mnemonics and 3 times for the three assembler directives (ORG, EQU, DN). If the program goes through the entire loop without finding a match, an ILLEGAL OP-CODE error results. If the program does find a match, the loop counter provides the op-code value of the mnemonic. When the program finds a STA, for example, the loop counter equals 3, the op-code value of a STA. Figure 1 shows a flow chart of this loop. Note that the variable names in this chart do not necessarily reflect the actual variable names in each version of *NanoAssembler*. We have chosen these names purely for explanatory purposes.

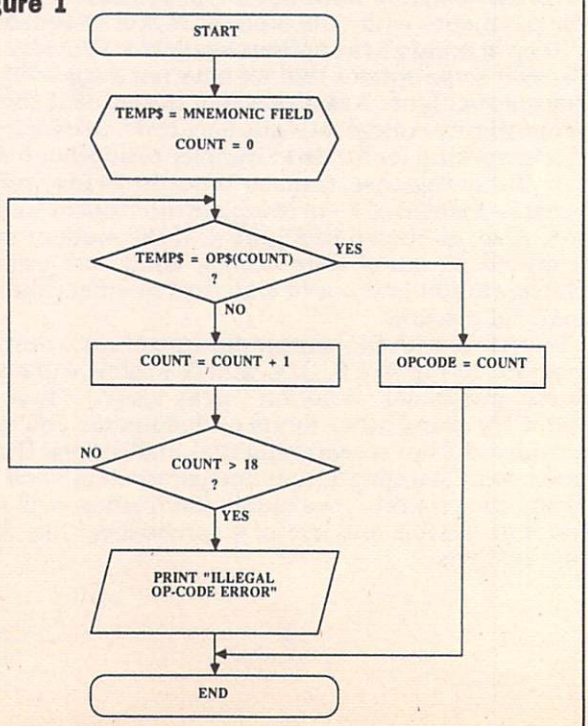
The Number Of Nibbles

Once the program has decoded a mnemonic into its op-code value, it must alot space for the op-code in memory. Not every op-code instruction demands the same number of nibbles. We therefore use another array—the NN( ) array (Number of Nibbles) to keep track of this amount. Set up in the same order as the OP$( ) array, this array stores the number of nibbles that each op-code instruction requires. For example, to find the number of nibbles that the STA instruction requires, you can print NN(3), with 3 representing the op-code value of a STA.

HCM Glossary terms: assembler, mnemonic, nibble, op-code.

HCM

Figure 1



HCM ONE-LINERS

Here they are . . . the best of the one-line programs that we have received since printing the fifth "HCM One-Liners" column in *Home Computer Magazine* Vol. 5, No. 5. Although many interesting programs were submitted, we have selected what we felt were the best 6 of those that arrived prior to this issue's press date (one for each brand of computer covered in our magazine, including a TI BASIC "10 liner"). If you have not yet submitted your masterpiece, it is not too late! As long as we keep getting great one-liners written in any computer language, we'll keep filling this page for you. Our prize winners this issue will each receive a check for $50 for sharing their ideas with our readers.

Pixel Palette
[TI Extended BASIC on the TI-99/4A]

Dear Sir:

My TI-99/4A one-liner lets you use a joystick to draw on the screen. You can design geometric configurations, letters, and numbers. To RUN this one-liner enter CALL CLEAR :: RUN. Make sure that ALPHA LOCK is up. The program displays your row and column position at the bottom of the screen. Type in the program until you hear the input beep, press [ENTER], then [FCTN 8], and finish typing the program in.

Robert Amenta
New City, NY

```
1 CALL CHAR(42,"FF9L/20R
),1SIJ4 THEN 7)(1C=31+XOS2)
L4OHEN C1+1,4):RPTS(2CAY>VHA
3 TR+1,C RPA1 R=FRHCH1
(R  C  IOLTO
```

Mandala
[Applesoft BASIC on the Apple II family computers]

Dear Sir:

This program creates oval patterns oriented around the center of the screen. With each plotted oval, the color alternates between black and white, forming an intricate design that covers the entire screen.

Andrea Sigurdson
Sidney, BC Canada

```
1 HGR2:FORD=1TO99:HCOL
0 R=.19:POKE-16384,RND(1)*6
79,190:FOR.=1TO99:FORG=0
=1TX=RND(1)*139:Y=RND(1)*9
:95B=FS+2:DRAW1,X,Y
*SA)TOB:CNEXTA:HCOLO
ECTOG:NEXT
```

Two By Two
[Commodore 64 BASIC on the C-64]

Dear Sir:

Here is a one-liner that allows limited word-processing. You can print two screen lines at a time, pressing [RETURN] after every two lines. You cannot use the comma, colon, or quotation marks. It works with most printers.

Chris Caldwell
Paris, TN

```
1 PRINTCHR$(14):OPEN3,4
:CMDRS:INPUT#3,A$:PRINT#3,G
OTO1
```

Sum It UP
[TI BASIC on the TI-99/4A]

Dear Sir:

My great 10-liner is called the *Add/Subtract Calculator*. When you first RUN the program, the "Totals" value is 0. You can add to this value by selecting 1 then entering the number you wish to add, or subtract from it by selecting 2 then entering a number. Press [ENTER] and the screen clears, showing the new "Totals" value.

Richard Lin
Moberly, MO

```
1 CALL CLEAR
2 PRINT CCEY(0,S,D)
3 CALL KEY(0,S,D)
4 IF (S<49)+(S>50)THEN
3
5 INPUT B
6 IF B=48 GOTO 7,9
7 C=C+B
8 C=C
9 G=G+1
10 GOTO 1
```

[NOTE: Because of built-in line length limitations in TI BASIC, we are now accepting "Ten-Liners" as entries for this column in a newly established TI BASIC category—Ed.]

Flying Fingers
[BASICA on the IBM PC, Cartridge BASIC on the IBM PCjr]

Dear Sir:

Here's a quick, little one-liner that will get your fingers flying in no time. *Qwiktype* lets you set the time for each practice session and keeps a running total of right and wrong answers as you type. It displays wrong answers as negative numbers. If you find that the program is displaying all negative numbers—no matter what you enter—hit [Caps Lock]. *Qwiktype* requires DOS.

Bob Langill
Hillsboro, OR

```
1 IF A THEN INPUT"TIF
GA(Y/N)";BS:IFBS="Y"THEN
"A=1:RANDOMIZE TIMERY="
S:INPUT"MINUTES";M:M=M*60
";IM,SB OCR:ONS)
GOSUB1:LD*26)+64.5)
HR$(RC,A$=INPUT$(1)
T D=D+1:C=C-(A$=C$)
:D
```

Topsy Turvy
[Atari BASIC on the Atari 800,800 XL, and 130 XE]

Dear Sir:

This unusual one-liner turns the characters on your screen display upside-down. It takes a while to RUN, but it's worth the wait.

Thomas Zaninovich
Pittsburgh, PA

```
1 POKE106,155:GRAPHICS0
:FORI=0TO1023:D=D+2*(DV
18)-16*(D=16):POKE39944+I
I:POKE756,156
```

All One-Liner submissions are subject to the same publishing criteria as Letters to the Editor (explained in the magazine's masthead, page 6). If you have written a great One-Liner in any language on any computer covered by *HCM*, send it addressed to: Letters to the Editor, 1500 Valley River Drive, Suite. 250, Eugene, OR 97401. You too may win a cash prize and be immortalized in print!

HOME COMPUTER

VOL. V NO. 6 ★★★★ INTERNATIONAL EDITION

Apple Attack

Corporate Image Tarnished As Firm Sues Ex-Chairman

Will Sales Of New Add-Ons, Drop In Holiday Prices Help Give Stock More A-Peel?

A tarnished corporate image, an unsteady financial picture, cutbacks, and legal saber-rattling now characterize the new Apple Corp. New product campaigns and improved fourth-quarter financials fail to wash away the spectacle of Apple suing co-founder Steven Jobs. Apple filed suit against Jobs—who split from the company he co-founded, taking 5 key employees—for breach of fiduciary trust. Meanwhile, Apple itself may be facing some litigation: Three publishers of recently released books about MacBasic may file suit against Big Red for cancelling the new language—and licensing the code to Microsoft to incorporate in its own version of BASIC for the Mac—prior to MacBASIC's long-awaited release. But the beleagured Apple is not deserting its new corporate image. Dropping its "Macintosh Office" marketing approach, the firm says it will stick with its already developed office peripherals. The new 20 megabyte hard disk drive, a forthcoming file server, and various third-party items linking Mac to the IBM PC could help carve a large enough specialized office niche. But Apple will also push its graphics-intensive Mac into vertical markets—such as desktop publishing. A new Mac keyboard, color monitor, "open chassis" Mac, Apple II/Mac cross-compliler, and reduced-price Laser printer are expected for 1986 debut. Among recently released products for the Apple II are a 3 1/2" disk drive for $299, compact 1200-baud modem for $399, and a combination color/monochrome monitor for $399. A memory expansion card that will increase the Apple IIe all the way up to 1 megabyte is forthcoming.

Ironically, the scuffle with Jobs and others may be drawing attention away from these new toys—just when the holiday season is hitting. Yet, Apple has cut the Mac 512K price by $300, bundled Apple II family products to create below-$1000 systems at retail, and revived its holiday ad campaign in a bid likely to lift Christmas sales.

"QUOTABLES"

"We believe the leading edge is often the bleeding edge in this business."

—James J. Edgette, of Entre Computer Centers, Inc., a retail chain that is reportedly not stocking Commodore's Amiga.

What's News—

* * *

Tandy Corporation's success with its $1000 PC-compatible family is spurring it on to smaller and better things—a 16-bit laptop machine with 3 1/2" disk for $1595. Watch for holiday bundling of color monitor with model 1000 at same $1000 price.

* * *

"Shrink Wrap" laws, like the one recently passed by Illinois, may put copy-protection "under wraps." Under such legislation, now being considered by several states, the act of opening a software package implies agreement to copyright conditions. Detractors fear this could weaken consumer warranties.

* * *

Digital Research buckled under Apple legal pressure to make its GEM format more "distinguishable" from Mac programs. IBM also ruffled DR's feathers by announcing it wouldn't sell GEM as earlier planned. Apple may threaten suit against other Mac lookalikes for "visual copyright" violations.

* * *

A complete encyclopedia on one compact laser disk by Grolier, Inc. will soon sell at $199 (for the IBM PC and Atari ST). Requiring a CD-ROM drive and interface ($500-1000), the medium provides super fast data search and full cross-indexing of its 9 million words—a real home computer advantage.

* * *

Old soldiers never die . . . TexComp has premiered a 128K 99/4A-compatible, while Franklin has announced a new Apple II-compatible with an operating system that reportedly meets Big Red's legal scrutiny.

* * *

Big Blue strikes again. IBM has pulled off a major coup—recently signing a deal giving it access to several hundred high-tech Japanese patents for advanced optical and chip technology. The deal stems from patent-licensing access IBM gave Japan after World War II, and is predicted to give IBM an unprecedented technological edge.

IBM Hanging Tough

PC Sales Hold Constant PCjr Gets Push

Pact With Microsoft Promises Stability

Sales of both the IBM PC and the XT are holding steady during the market shakeout, while Big Blue has reaffirmed its commitment to the PC line's open architecture—recently agreeing to continue the reign of Microsoft's PC-DOS, rather than revert to a once-rumored proprietary operating system. This is good news for both third-party developers and those who use their products. Some skeptics think this is merely a ruse to buy time until IBM can issue a new closed-DOS 80386-based machine. The facts suggest, however, that IBM will probably have Microsoft revamp their TopView software, and fold MS-Windows into it.

Inclusion of a 30 Mbyte hard disk on the AT for a mere $200 increase in cost over the 20 Mbyte "garden variety" version may foreshadow yet another round of price cuts for all IBM personal computers. Meanwhile, expect another PCjr Christmas. With the street price of Junior with color monitor around $599 and special promotional offers to schools, the estimated 300,000-unit inventory should be cleaned out by early spring—just in time for the possible U.S. introduction of the Japanese-introduced (and Australian test-marketed) JX model with 3 1/2" disk drive.

Commodore Struggles

Firm Still Clinging To Amiga's Potential Success

But Will New Model Be A True Friend?

Commodore is suffering. Doomful company predictions foretell an $80 million loss in the firm's fourth quarter, a layoff of 700 employees, and a $50 million inventory write-down. Commodore executives are counting on Amiga to bring the company out of its slump. The firm hopes that the new computer, billed in a scheduled $40 million dollar ad campaign as the "Creative Edge," will return the company to solid financial ground.

But Amiga may not be enough. Analysts speculate that Commodore's home market image, its unreliable reputation, and lack of distribution will be impossible to overcome—even though a national ad has listed 662 dealers ready to handle the new computer. And Amiga has problems of its own: Its late arrival and lack of software development—in part due to an incompletely defined operating system—weaken its holiday appeal to all but a dwindling supply of "early innovators."

Commodore's unexpected success in dealer sign-up may be attributed to its single-level pricing structure and lack of purchasing volume requirements—a lure that allows stores of all sizes an equal chance to compete, while discouraging gray marketeering.

INDUSTRY JOURNAL™

© COPYRIGHT 1985 EMERALD VALLEY PUBLISHING CO. EUGENE, OREGON NO CENTS

Trouble In Atari-Land?

Two Top Execs Make Tracks Firm's Fate Rests On ST

Will Apple Grab Jackintosh By The GEMs?

Dealers complain of too-tight sales conditions, others worry about lack of software, still others wonder if Atari's funds will hold out—but Jack Tramiel keeps touting in print ads his $799 520 ST—dubbed the "Jackintosh"—as the "Rip-off Buster." Atari's future hangs on this new machine—for which it now claims to be shipping a BASIC language after long delay. With uncertain cash reserves, the firm doesn't seem to have much more time. Meanwhile, amid rumors of mismanagement by Tramiel, Atari lost two top executives.

But there is a new question: Will Macintosh sink Jackintosh? The answer may lie more in legalities than kilobytes—if Apple brings its legal guns to bear. Atari's 520 ST earns its nickname from its Mac-like GEM environment. By its settlement with Digital Research, the developers of GEM, Apple has forced major changes to GEM Desktop, Draw, and Paint. A similar suit against Atari might delay the ST enough to empty company coffers. Sam Tramiel says, however, that DR has only agreed to "certain superficial changes"—and that, "Contrary to reports, the agreement does not require changes to GEM [itself] or to Atari Corp.'s TOS operating system."

Software Market Trends

Firms Merge Drop Protection, Prices

Site-Licensing The Norm In Corporate America

Consolidation, lower prices, less copy protection, and more site licensing typify the now-more-mature software industry. Borland International is a case in point: Immediately after Borland recently acquired Analytica, makers of the Reflex database, it slashed Reflex's price—from $495 to $99—and eliminated its copy protection. Borland will now also offer site licensing for its best-selling Sidekick desk accessory software package, whereby—for $50,000—large corporations can make unlimited program copies.

Cyber Communications

High-Speed Modems Head Movement

Are Sci-Fi Videophones Just Over Horizon?

New high-speed modems of 2400 baud and beyond, more powerful telecommunications chips, and network-creating alliances like the recent IBM and MCI merger are laying the groundwork for the long-awaited communications revolution.

IBM is working toward a full voice, video, and data distribution network—perhaps leading to a method of receiving voice-read data from a personal computer over the phone. Epson has teamed with ABC to send electronic mail over FM radio. Meanwhile, go-it-alones like GTE with its Sprint service are expected to be at a severe competitive disadvantage.

Printshops Beware

Desk-Top Publishing Becomes Viable Alternative

Explosion of Lasers, Versatile Pagination Software

Computers with super printers now make it possible to publish near print-quality material from a desktop. Full-up systems with adequate memory and powerful software are already here—at prices that are convincing large numbers of firms to bring the function "in house." Analysts expect rapid hardware advances in resolution, color, and speed, plus software featuring auto-hyphenation and easier-to-use "human interfaces." Newer printer technologies will slug it out with the now-dominant laser print engine. Ricoh is rumored to be introducing a five page per minute, under-$1000 machine by year end.

THROUGH THE LOOKING GLASS

Artifical Intelligence Gives Expert Advice As Machines Gradually Become More Human

Computer, computer, on the wall—what's the best choice for us all? If *your* computer could make decisions for you, would you consider it intelligent? Would you settle for some good advice? Some of us may expect much more—like self-awareness and creativity—from a truly intelligent machine. But at the current state-of-the-art, what we can get is *expertise*. Knowledge-based or "expert" systems are popping up all over, finding use in management, on the assembly line, and in ever more applications for personal computers. These software systems provide advice in reponse to real-world questions like "What do I do now? The thingamajig won't budge!" Instead of a whole set of instructions, you get the kind of measured, specific advice you would expect from an engineer looking over your shoulder: "That doesn't work? Try this . ."



But say you want to discuss the meaning of the universe with your computer. You may find that the machine is not so dumb as it is *ignorant*. Expert systems are limited by processing speed, memory, and storage capacity—so naturally the average 128K home computer will not provide very quick decisions, or have much to base its advice on. Most useful systems need megabytes of data in memory and on hard disk or CD-ROM. They must be able to search through an immense set of alternatives for every possible situation in the knowledge base, then narrow all these down to the correct choice. Obviously, the universe is a *very* immense set of alternatives for *anyone* to handle—including the smartest computer. But "full-up" systems consisting of a personal computers with super expansion cards, hard disk drives, and CD-ROM capability are quickly becoming more affordable. In situations that demand specific expertise, such as fixing the widget on your automobile, artificial intelligence is coming home.

As we build smarter systems , human-like intelligence may yet come to machines—perhaps to make many important decisions for us. *Now* may be the best time to start deciding what decisions we will want them to make.

Soundbytes

Of Music and Microchips

by Andy Widders-Ellis
HCM Staff

Hark! Does the sound of electronically-produced music herald the next stage in the evolution of the personal computer?

It's a fact: Personal computers have permanently changed the way people produce music. The impact of this technology can be heard all around us. Curiously, the other side of the coin is often overlooked—how music has stirred up the world of home computers . . .

Heresy!

Music has been around a *long* time, and—let's face it—humans would have continued to express themselves through sound with or without the electronic revolution of the 80s (albeit in much more traditional ways). Music *benefits* tremendously from the home computer, but doesn't *require* it to stay vital.

On the other hand, what about the field of home computing—hasn't evolution been necessary for some time now? How many personal computers are gathering dust because their owners lack some deeper, satisfying long-term involvement with them?

Home computers have the reputation of drawing the user into isolated, solitary activities. We all know the typical stereotype: A computer user spends a lot of time alone—apart from family and friends—playing games, keeping books, studying, word processing, etc. Of course, most of us need some time alone, simply for peace of mind and mental health. An activity that allows us to work in reflective seclusion can be valuable.

But how much *more* valuable is a hobby that promotes both private time and *together* time? Computer-assisted synthesizer programming, composing, arranging, sequencing, and editing *does* necessitate working by yourself. Yet, once a creative project is complete, the musical fruits of this labor practically demand to be *shared*. Ultimately, music deserves an audience—the obvious thing to do once you've created your music is to let someone hear it.

Out Of The Closet

We are witnessing a renaissance: Once used as fancy calculators, game players, and effective but utilitarian business machines, personal computers are now becoming creative tools for personal self-expression. Thanks to this creative potential, *inspiration* for working with home computers is re-kindling. Home computers are emerging from the closet to appear in the living room again. Why? Because folks want hobbies and pastimes that they can develop over a lifetime, and that yield useful skills in exchange for the energy invested. To date, few computer activities have offered this much long-term, open-ended growth potential.

Also contributing to the rejuvenation of the home computer is the new owner who was never interested in mere "computing." These first-time buyers—at least initially—have one application in mind, namely music. It will be interesting to see whether, in time, these people find other uses for their computers.

What's It All About, PC?

If you subscribe to the theory that many of mankind's noblest ambitions and achievements are in the realm of *communication*, then something that supports this effort becomes very precious. Since the dawn of civilization, music has manifested this power to bring people together. We know man-made sounds can heal or wound, create or destroy, excite or soothe, elevate or depress. Cruise the radio dial to confirm this!

The point is this: Home computers place this powerful means of self-expression—once monopolized by a trained elite—into more hands than ever before. If you own a home computer, then *you* can wield previously unheard of control over this potent force. Doesn't this realization make the thought of playing "just another video game" seem a tad tame?

Knock On New Doors

The old notion that musical expression is for a chosen few no longer applies. With today's technology, you can compose and sequence wonderfully complex and detailed music with only *one finger*! If you have musical ideas, they can be realized using a home computer and computer-based electronic musical instruments to carry out the *physical* task of real-time sound production. Computers force us to redefine the term "musician."

Musical self-expression is within your grasp, even if you are a novice. Sure, you have to work at it—creative expression takes persistent effort. But the rewards justify this effort. As you gain knowledge and experience in music, you will come to see both yourself and your computer in a new light. Self-expression is satisfying and liberating, as therapists will attest. And the instrument that facilitates this creativity becomes especially important to the user.

A new generation of computers is emerging, combining greater sophistication, super hi-res graphics, and more power—with lower cost. These features make them ideally suited for use as creative tools. The new Atari 520 ST even comes with a built-in MIDI (*Musical Instrument Digital Interface*) port! Obviously, Atari is counting on musical applications to enhance the value of its new product.

Turn On, Tune In, Take Off

As our acknowledged universal language, music has accompanied (and often shaped) the unfolding drama of man. Why not come "on line" with your own contributions? A home computer can be your ticket to this creative journey. Who knows what you'll discover?

HCM Glossary terms: MIDI. **HCM**

MUSIC & ELECTRONICS™

The Magazine of Creative Discovery Through Sound Technology

VOLUME 1 NUMBER 1

$3.50 in U.S.A.

THE NEW MUSIC MAKERS

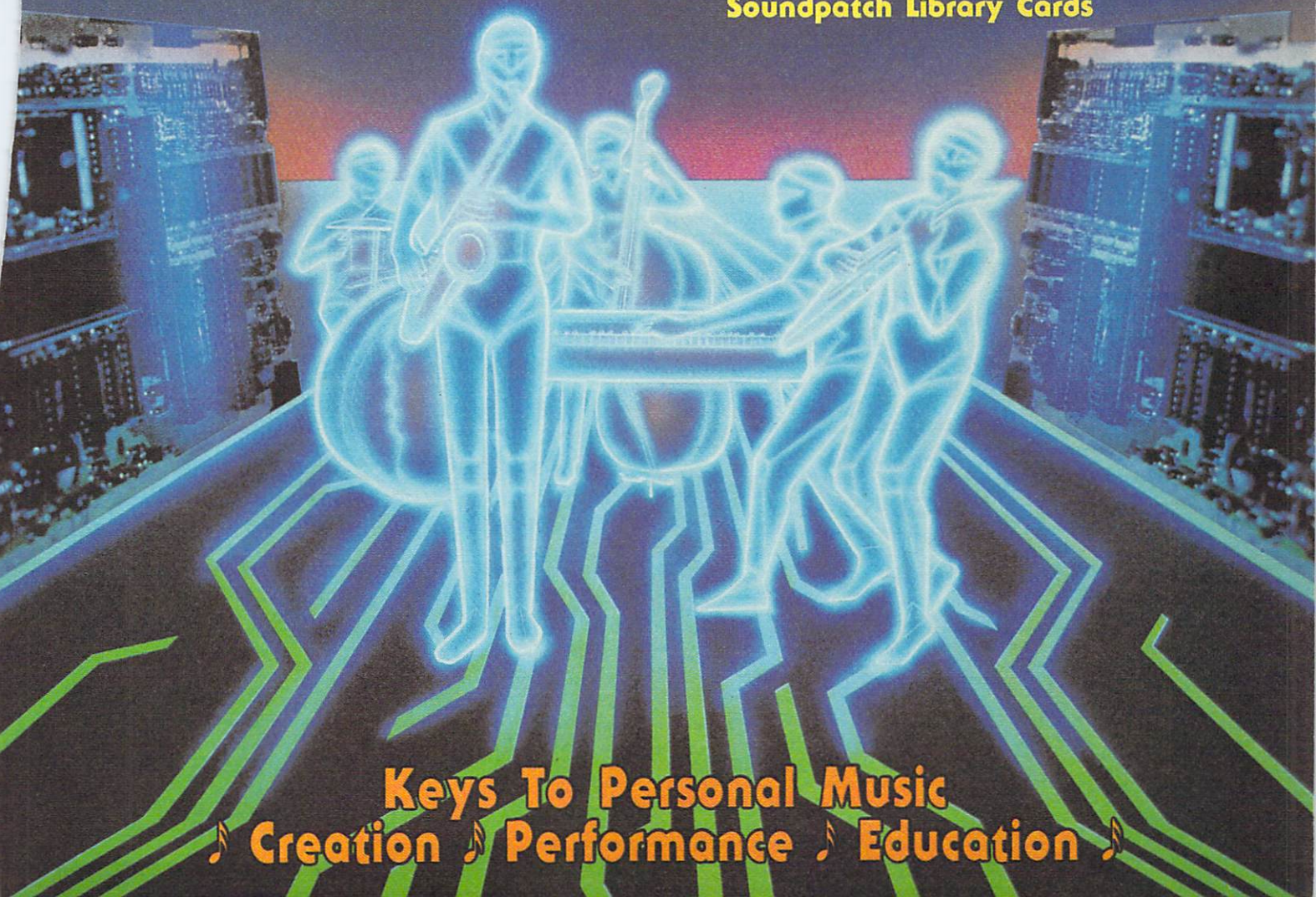
The Compact Home Music Studio

Home Computers: New Dimensions in Musical Participation

Synthesizer Soundings & Digital Orchestration

Stretch Your Tape Recording Dollar:
Affordable Multi-Track Cassette Equipment & Techniques

Featuring a Bound-In Sound Demo Record
PLUS—A Free Set of Soundpatch Library Cards



Keys To Personal Music
♪ Creation ♪ Performance ♪ Education ♪

MUSIC & ELECTRONICS™

The Magazine of Creative Discovery Through Sound Technology

Music & Electronics is devoted to "the musician in all of us." Each issue explores the many facets of personal music creation, performance, and education through the magic of computers and electronics. Editorial content spans a full range of subjects—from creating the Electronic Orchestra by interfacing musical devices with home computers, to analog and digital recording techniques, to computer-assisted music lessons. Bound into each issue is a Soundsheet demo recording, adding an extra dimension to magazine content, and an assortment of Soundpatch Library Cards for re-creating unique sounds on current musical devices. **Each 1-year *Music & Electronics* subscription includes 6 magazine issues, plus 6 companion audio cassette tapes which provide in-depth audio examples directly relating to the magazine's tutorial articles and in-depth reviews.**

Electronic Music Lab™
Basic Principles of Music Synthesis
What is this thing called sound? What distinguishes one sound from another? How is sound created electronically? We answer these and many more questions in this basic primer on the physical phenomenon of sound and synthesis. Our clear explanation of this medium takes the mystery—but not the wonder—out of the new musical technology.

Synthesizer Soundings™
Hands-on Lessons
Here we demonstrate how anyone can "program" the most popular synthesizers. It's not just a matter of moving switches and pushing buttons. M & E's mandate is to teach the art of sound design.

The Digital Sampler™
From Physics to Numbers and Back
Digital recording is a fundamentally new way of capturing sound. Home computers can play a role in handling and manipulating this musical data. Learn how to work with this new technology, as M&E takes you along the digital path.

The M&E Soundpatch Library™
Ready-to-Use Synthesizer "Patches"
Looking for new and useful sounds on your electronic instrument? Each issue contains easy tear-out cards with original patches for popular synthesizers. When appropriate, soundpatch data from the magazine will also be available on tape or disk.

Computer Home Companion™
Home Computers and Music Magic
Home computers—some of which have small built-in synthesizers—are opening up the floodgates to popular participation in the creation and performance of music. Some software teaches music theory—while other programs convert the computer into an instrument itself. M & E also shows readers how computers can record and control musical devices.

The Home Music Studio™
The Art of Analog & Digital Recording & Mixing
Miniaturization and cost-reduction of electronic instruments, computers, and recorders have made compact, affordable home studios possible. In this series, we help you develop your own recording system—providing tips on design, construction, and operation, as well as advice on selecting the proper equipment. Learn multi-track and MIDI recording techniques, signal processing tricks, and how to achieve professional results from your personal studio.

M&E Rhythm Beat™
Programming Digital Drum Machines
Drumming has changed radically since Ringo struck up the band. Today, even those who never mastered the sticks can program a machine to both imitate drummers and play humanly impossible "licks." The digital drum machine is an extension of the traditional drum set—for which rhythmic skills are essential. Learn the rudiments of rhythm; enjoy lessons in style; explore the world of the big beat.

MIDI Melting Pot™
Plugging into the Industry's Common Communications Standard
MIDI (Musical Instrument Digital Interface) is an industry-standard interface that sends musical information in digital form between synthesizers, signal processors, and computers. This interface can link many electronic music devices in one large ensemble, or connect the synthesizers and signal processors to a computer for digital recording and editing. Learn how MIDI is indeed the key to the Electronic Orchestra.

Composer's Workshop™
Creating & Arranging a Musical Idea
Electronic instruments make it easy to write and arrange music, even for those with little or no previous experience. Let M & E show you how to generate and develop ideas—bringing your music out of the shower, into the streets.

In Session™
The Making of a "Track"
Listen to the bound-in sound recording, then follow us as we document the entire recording and mixing process step-by step, using text, diagrams, and color photographs.

Pro-Formance Playbill™
M & E's beat covers the exciting world of electronically-produced music through interviews with major artists plus reviews of concerts and new releases.

M & E Review Roundup™
M & E provides incisive reviews of electronic musical instruments and equipment plus in-depth analysis of the latest software products.

Ask M&E™
Here our readers get to sound-off with any questions related to electronics and music.

And There's More:
- **Create-A-Sound Contest™**
- **Compact Disk Corner™**
- **New Products Soundcase™**
- **Notes to the Editor™**
- **M & E Glossary & Abstracts™**

—Plus a few special surprises . . .

YES! Enclosed is $32.97 (Canada: $39.97) in U.S. funds. Please enter my charter subscription of 6 magazine issues and 6 companion ON TAPE™ cassettes to

MUSIC & ELECTRONICS™
The Magazine of Creative Discovery Through Sound Technology

Please Print
Name _____
Address _____
City _____ State _____ Zip _____
☐ Check or Money Order Enclosed
MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK
Charge my ☐VISA ☐MasterCard
Account No. _____
Date Expires _____
Tel. No. _____
Signature _____

Clip or photocopy this coupon or use the bind-in order form at center of magazine.

★ **AN AUDIO-VISUAL SPECIAL** ★
For Charter Subscribers*
DON'T MISS OUT ON THIS EXTRAORDINARY SUBSCRIPTION VALUE!

SAVE 42%

Featuring:
- *6 magazines PLUS 6 companion ON TAPE™ cassettes*
- *Unique performances from the latest electronic musical instruments*
- *Demonstrations of step-by-step recording techniques*
- *Lessons on creating original synthesizer sounds*
- *An audio odyssey through M & E's popular columns: "Composer's Workshop" and "In Session"*

The Vol. 1 No. 1 Premier Issue of magazine and cassette tape will be mailed in December

Enclose payment or credit card information & mail with completed form to:
Emerald Valley Publishing Co. ● P.O. Box 70288 ● Eugene, OR 97401
Or use our TOLL-FREE Order Line for VISA/MasterCard orders only:
1-800-828-2212 In Oregon, Alaska, Hawaii Tel. (503) 485-8796

Satisfaction Guaranteed—or the unfilled portion of your subscription will be refunded.
Prices subject to change without notice.

*NOTE:
If you have already subscribed to the magazine *alone* (no longer offered), we extend to you the opportunity of adding the difference ($14.97 in U.S.) in charter rates to convert over to the full audio-visual companion package of 6 ON TAPE™ cassettes—for an additional $15 savings!

HOME COMPUTER® product news

Each month we publish items of interest and news of recently or soon-to-be released computer products. Our publication of information from manufacturers of computers, peripherals, software, and accessories is not to be construed as product endorsement. Prices quoted are the manufacturers' suggested retail prices and are subject to change.

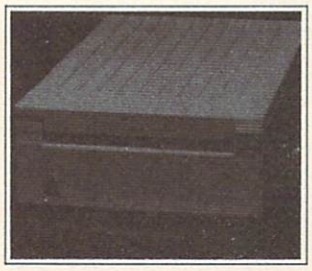
Send press releases to:

Product News Editor
Home Computer Magazine
1500 Valley River Drive., Suite 250
Eugene, OR 97401

❖❖❖❖

Add A Data Warehouse

New Apple II Drive Stores 800K



Apple's new UniDisk 3.5" floppy disk subsystem substantially expands the data-storage capacity of the Apple II+, IIe, and IIc. The system requires 3.5", double-sided disks that provide 800K of storage. Unidisk can be used in conjunction with other Apple II drives. UniDisk also loads and saves data 50 percent faster than conventional Apple drives. It connects directly into the back of the IIc or into an interface card for the, II + and IIe. The disk drive is available for $499. (UniDisk on an Apple II + and IIe, requires an Apple IIe Accessory Kit, which sells for $69.)

Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, CA 95014
(408) 996-1010

❖❖❖❖

Courtroom or Castle?

Two New Adventures Released

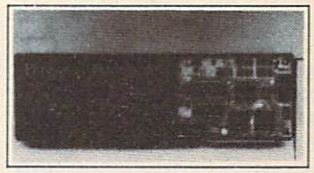
Telarium recently released two interactive adventure games. In Perry Mason: The Case of the Mandarin Murder, users play the role of Perry Mason, performing all the duties of a real courtroom lawyer (cross-examining, introducing evidence, and consulting privately with the judge). In Nine Princes in Amber, users try to capture the throne of Amber. Negotiation, politics, and alliances are key in this game. Commodore 64 and 128 versions are available for $32.95. IBM PC, PCjr, and Apple II family versions are $39.95.

Telarium Corp.
One Kendell Square
Cambridge, MS 02139
(617) 494-1200

❖❖❖❖

Just Plug it In

PC Hardcard Boosts Productivity



Plus Development has introduced the first hard-disk drive on a plug-in board for the IBM PC. The Hardcard is designed to help PC users achieve the productivity gains of hard disks without losing their current investment in the PC. It comprises a 10-megabyte disk drive and installation software on a plug-in card (measuring 4" x 13" x 1"). Hardcard does not require the repeated use of installation software and comes with Hardcard Directory, a program that allows users to locate application programs and data files on their hard disk. Hardcard will be available for a suggested retail price of $1,095.

Plus Development Corp.
1778 McCarthy Blvd.
Milpitas, CA 95035
(408) 946-3700

❖❖❖❖

What's the Capital of . . .

Educational Software for the 99/4A

Trinity systems has introduced USA States and Capitals, an educational game for the TI-99/4A. The object of the game is to identify more states and their capitals than your opponent. Players win points for each correct letter they type in as part of their answer. The game also has a Tutor Mode, which indicates each state's location on the map, then displays its name, capital, and nickname. The cassette game runs in TI BASIC and requires no additional software or memory. It retails for $12.

Trinity Systems
1022 Grandview Ave.
Pittsburgh, PA 15237
(412) 366-5811

❖❖❖❖

Mount-In Mega Memory

Expansion Card For Apple II Family

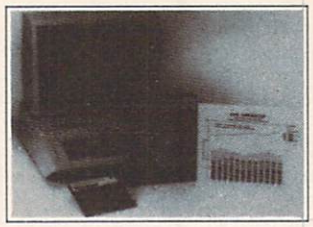
Apple plans to soon release a new Apple II Memory Expansion Card for the Apple II, II + , and IIe. The card expands memory in 256K increments to over one megabyte, permitting larger files. The card includes RAM, ROM, and a custom integrated curcuit. The Memory Expansion Card's suggested retail price has not yet been announced.

Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, CA 95014
(408) 996-1010

❖❖❖❖

Color Plotter "Solution"

Keychart Gets to Core of Apple Graphics and More



Softkey Software Products has rebuilt Keychart—a business graphics program—to operate on the Apple II, IIc, and IIe. The Apple-compatible program features on-screen layout preview capabilities and supports more than 20 different plotters and printers, including those manufactured by Hewlett Packard, Houston Instruments, and Apple. Keychart produces combination bar, line, symbol, pie, X-Y, horizontal bar, and scatter charts from data compiled on Lotus 1-2-3, Symphony, SuperCalc, Visicalc, MultiPlan, and Framework. Versions of Keychart are also available for the IBM PC, AT, XT, PCjr; IBM compatibles; and Texas Instruments Professional. It sells for $375.

Softkey Software Products, Inc.
18480 Decatur Dr.
Monte Sereno, CA 95030
(408) 395-1974

❖❖❖❖

Four Years in the Wings

Super Pascal Developed for Commodore

Abacus Software has released a complete Pascal Development System—Super Pascal—for the Commodore 64 and 128. Super Pascal utilizes a file editor, an assembler for coding-in machine language routines, a utility package for file and disk management, high-precision 11-digit arithmetic, and high-speed DOS. The package, which includes a 200-page handbook, sells for $59.95.

Abacus Software
P.O. Box 7211
Grand Rapids, MI 49510
(616) 241-5510

❖❖❖❖

Carrying a Heavy Load

Big-Byte Disk Drive for Atari

Astra System's newest disk drive, the Big D, is a double-sided, single- or double-density dual disk drive for Atari computers. Big D is supplied with the latest version of TOPDOS and can store 720K. It's compatible with Atari's new XE Series, as well as Atari's intermediate computers—the 400, 800, 600XL, 800XL, and 1200XL. Astra also makes two other Atari-compatible disk drives—the 2001 and the 1620. Both are supplied with SMARTDOS or MYDOS and can store 360K. The Big D has a suggested retail price of $595. The 2001 is $495, and the 1620 will be closed out at $300.

Astra Systems, Inc.
2500 Fairview, Unit L
Santa Ana, CA 92704
(714) 549-2141

❖❖❖❖

Hang On To Your TI

Software Combines PC and 99/4A

Intelpro has introduced Upwards!, a new product designed to quickly transfer data between the TI 99/4A and the IBM PC. The package includes two programs that run concurrently on both computers, so that the TI serves as a "remote file server," transferring data at a rate of 8,000 characters per minute. The product requires a fully configured TI 99/4A, an Extended BASIC cartridge, an IBM PC with at least 128K of memory (256K is recommended), DOS 2.0 or 2.1 for the IBM, RS-232 capabilities for both computers, and a cable to connect the two computers. Upwards! has a suggested retail price of $79.95 for U.S. buyers and $99.95 for Canadian buyers.

Intelpro
5825 Baillargeon St.
Brossard, Quebec, Canada J4Z 1T1
(514) 656-8798

❖❖❖❖

MIDI Meets TI

New World Class MIDI Software For 99/4A



World Class Software unveils TEX-SEQ, a MIDI (Musical Instrument Digital Interface) sequencer package for the TI-99/4A computer. TEX-SEQ comes complete with MIDI cable interface and software. The package allows note-by-note entry of musical compositions, which will play on any MIDI synthesizer. Software provides 4 play tracks and supports several different time signatures. TEX-SEQ requires 32K memory expansion, Extended BASIC, and a disk drive. Retail price is $49.95.

World Class Software
1500 Valley River Dr., Suite 250
Eugene, OR 97401
(503) 485-8796

❖❖❖❖

Homework Helpers

Bring the Teacher Home

Spinnaker Software unveiled Homework Helper Math World and Homework Helper Writing to teach students in grades 7-12 step-by-step methods for math and writing homework. Math World contains tutorial and problem-solving sections, and Writing focuses on book reports, essays, and research papers. Both programs are available for the Commodore 64 and 128 at $32.95 and for the IBM PC and Apple family computers at $39.95.

Spinnaker Software
One Kendell Square
Cambridge, MA 02139
(617) 494-1200

❖❖❖❖

Commodore Cost Calculator

Crunch Bills With Energy Manager

A new program for the Commodore 64, designed to track energy use in homes and buildings, is now available from Powerline Software. Energy Manager uses utility bills as a data base to analyze heating and cooling energy usage. It enables month-by-month comparison of energy expenditures for the current year with those of previous years and measures the effect of energy conservation tactics. Energy Manager comes on diskette with interactive data-entry and editing programs, analysis programs, sample data, and user's manual. It's available for $59.95.

Powerline Software
P.O. Box 635
New Hartford, NY 13413
(315) 735-0836

❖❖❖❖

Hands-On Screen System

Apples Get the Personal Touch

Personal Touch is now shipping Touch Window, a four-in-one touch screen input device for the Apple II family of computers. Mounted directly on a monitor, Touch Window provides any Apple II family computer with a see-through touch screen system. The package comprises a word processor, a picture/graphic creator, a spreadsheet application, a puzzle, and a program to check and recalibrate the touch screen. The product can be removed from the monitor for use as a graphics tablet, input pad, or interactive book pad. The system sells for $199.95 and comes with the Interactive Book I, a book containing touch screen applications and activities for ages three to adult.

Personal Touch Corp.
4320 Stevens Creek Blvd. Suite 290
San Jose, CA 95129
(408) 246-8822

❖❖❖❖

A First for Forth

Apples Speak Same Language

MicroMotion recently released Masterforth for the Apple family of computers. This version of the Forth programming language provides a complete programming environment for the Apple family, comprising a 65C02 macro-assembler and full interface to Apple DOS 3.3. As standard features, MasterForth employs a string package, screen editor, and resident debugger. Version 1.0 retails for $125 and is also available for the Macintosh, the IBM PC and CP/M-operating system machines. Users can write software on one system and run it on all the others.

MicroMotion
12077 Wilshire Blvd. #506
Los Angeles, CA 90025
(213) 821-4340

❖❖❖❖

Energized Word Processing

PaperClip is Revitalized for Atari, Commodore Computers



Originally issued for the C-64, the PaperClip Professional Word Processor by Batteries Included is now available in an updated version for Atari 800 and compatible computers. The program offers Mail-Merge, dual-editing windows, graphic capability, and many printing functions. It sells for $80. PaperClip has also upgraded the Commodore version with a SpellPack. Its dictionary includes over 20,000 entries and can be user-expanded by 5,000 words. PaperClip with SpellPack for the Commodore retails for $149.

Batteries Included
30 Mural St.
Richmond Hill, Ontario
L4B 1B5 Canada
(416) 881-9816

❖❖❖❖

TI Clone With A Plus

Tex-Comp Unveils 128K 99/4A-Compatible, Peripherals



Tex-Comp has announced its latest development, the TC-99/9. It's a 128K TI-compatible computer with a TI processor board modified for a 40/80 column display. Tex-Comp does not plan to produce the TC-99/9 except on a special order basis. The firm expects to sell the computer for under $1500. Tex-Comp also introduced its latest in enhanced peripherals for the TI-99/4A: the TC-1, TC-2, and TC-3. The TC-1 is a new expansion enclosure equipped with a fan-cooled regulated power supply, two double-sided, double-density drives, and a rear panel of 6 spike and surge protected accessory outlets. It sells individually for $450 and, packaged with the Corcomp 9900 Micro Expansion, for $700. The TC-2 includes the TC-1, the TI speech synthesizer, the Corcomp 9900, and the the TI-99/4A console. Available by special order only, it sells in the $800 range. Also supplied by special order is the TC-3—containing the same electronics as the TC-2, but with an added 128K of memory. It sells for under $1000.

Tex-Comp
P.O. Box 33084
Granada Hills, CA 91344
(818) 366-6631

❖❖❖❖

Black Belts Beware

Games Converted for Atari

Broderbund's Championship Lode Runner and Karateka are now available for the Atari 400, 800, and XL/XE computers. In Lode Runner, players attempt to elude guards and move a galactic commando across floors to retrieve gold. Karateka is a karate simulation, wherein users play the role of a young karate master who must battle with an evil warlord and his warriors to rescue a princess. Karateka for Atari computers requires a joystick and 48K. Lode Runner may be played with a joystick or keyboard and also requires 48K. Both games are $34.95.

Broderbund Software
17 Paul Dr.
San Rafael, CA 94903
(415) 479-1170

❖ ❖ ❖ ❖

Software Cloning

MIDI Music Shop Developed

Passport Designs has entered into a licensing agreement with Broderbund Software and developed a MIDI version of Broderbund's The Music Shop. Passport reviewed many music-composing programs and chose The Music Shop for its editing features. The software allows users to create, store, and edit compositions and print sheet music in piano or single-staff (treble, bass, or both) formats. It requires a one-drive Commodore 64, a MIDI keyboard, a joystick, the Passport MIDI Interface, and a dot-matrix printer. The MIDI version of The Music Shop retails for $99.95 and is available at music stores.

Passport Designs, Inc.
625 Miramontes St.
Half Moon Bay, CA 94109
(415) 726-0280

❖ ❖ ❖ ❖

Quantum Leap for Commodore

Telecommunications Network Introduced

Quantum Computer Services has announced the start up of QuantumLink, a nationwide interactive telecommunications service for Commodore computers. The network offers software previews, electronic messaging and bulletin boards, teleshopping, news, a computer information center, interactive telegaming with full-color graphics, on-line chat, and an electronic encyclopedia. QuantumLink's monthly subscription fee of $9.95 includes connection fees and communications charges for most of the offered services. In early 1986, QuantumLink will be available to other computers.

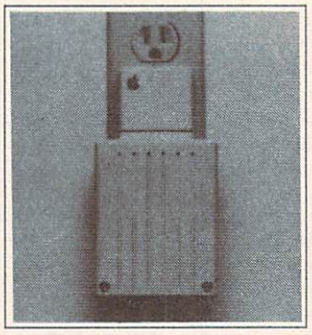
Quantum Computer Services, Inc.
8620 Westwood Center Dr.
Vienna, VA 22180
(703) 448-8700

❖ ❖ ❖ ❖

Modem, Modem On The Wall . . .

Telecommunications With The Apple II

The Apple Personal Modem, now available from Apple, is a compact 1200/300-baud modem that plugs into a wall socket or power strip and enables any Apple personal computer system to communicate with mainframes, minicomputers, and other microcomputers. The Apple Personal Modem works with either tone-generating or pulsed telephone systems. It has dial, answer, and status-display capabilities. The Apple Personal Modem is available for $399.



Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, CA 95014
(408) 996-1010

❖ ❖ ❖ ❖

The Next Generation of Printers

Epson Introduces FX New-Borns

Epson America recently rolled out two new printers designed for business applications—the FX-85 and FX-185. Both models produce draft copy at 169 cps and near-letter-quality copy at 32 cps. They also offer single-stroke-selectable printing modes from built-in IBM character sets and an 8K print buffer that frees computer time while printing. The FX-85 is listed at $499, the FX-185 at $699. Optional cut-sheet feeders for automatic single-sheet use are available for both models ($269 for the FX-85 feeder and $319 for the FX-185 feeder). Epson also offers $85 upgrade kits for older FX-series printers.

Epson America
2780 Lomita Blvd.
Torrance, CA. 90505
(213) 421-5426

❖ ❖ ❖ ❖

It's Only Logical

Software Designed to 'Booleanize' Students

Sunburst Communications recently made High Wire Logic available for the Apple II family of computers. Originally part of Sunburst's Discrimination, Attributes and Rules package, the software introduces students to Boolean logic and strengthens and extends students' capabilities in logical thought. The program is designed for students in grades 5-12 and retails for $59. It requires a 48K Apple II-series computer.

Sunburst Communications, Inc.
Pleasantville, NY 10570
(914) 769-5030

❖ ❖ ❖ ❖

HOME COMPUTER™

PROGRAM LISTINGS

CONTENTS

Software Instructions Page No.	Article	Page No.	Page No.	Page No.	Page No.	Page No.
15	TI Card Shuffler					
	Listings					108
	Programmer's Window					76
16	Electronic Postman					
	Listings	97	99	102	105	
	Programmer's Window	78	80	77	79	
19	Serf City					
	Listings	109	112	114	117	120
	Programmer's Window	83	85	82	84	81
22	Cell Mates					
	Listings	122	124	126	127	129
	Programmer's Window	88	90	87	89	86
25	NanoAssembler					
	Listings (NanoAssembler/Editor)	131	134	137	140	142
	Programmer's Window	93	95	92	94	91
61	HCM One-Liners	61	61	61	61	61
52	Apple Seedlings	53				
44	Atari Atrium		45			
46	Commodore Hornblower			146		
50	IBMpressions				51	
54	TI Razzle Dazzle					55
42	Apple Tech Note	42				
43	Atari Tech Note		43			
48	C-64 Tech Note			49		
56	IBM Tech Note				NA	
57	TI Tech Note					NA

Apple Note: HCM Programs RUNing on the Apple II+ require 64K.
Atari Note: All HCM programs RUN on the Atari 800, 800-XL, and 130-XE. Shorter HCM programs should also RUN on other Atari models with less memory.
Commodore Note: All HCM Programs also RUN on the C-128 in C-64 mode.
Tandy Note: All programs for the IBM PC/PCjr commencing with the Vol. 5 No. 5 issue are designed to RUN without modification on the Tandy 1000.
Franklin Note: See page 9.

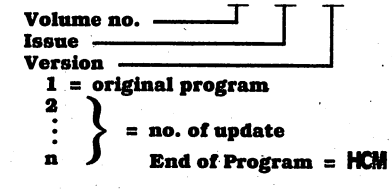
WHAT IS A PROGRAMMER'S WINDOW?

Home Computer Magazine strives to serve those eager to expand their programming knowledge by including specific information about the software published in these pages. This information is contained in a separate section: the "Programmer's Window." For each main program included in this section there are 5 different pages—one for each brand of machine we cover (Apple II family, Atari 800-compatible family, Commodore 64, IBM PC & PCjr, and TI-99/4A). Each page contains 4 separate windows as follows:

1) **Design Focus** (flow chart or diagram of a specific procedure or program structure)
2) **Remarks** (text explaining an aspect of the program version)
3) **Directory of Variables** (definition of all variables)
4) **Listing Annotations** (line-numbered program guide)

Program Identification

Each program header (the first few lines of the program) contains information giving the language the program is written in (e.g., TI Extended BASIC, Applesoft, etc.) and any special system components that are required (memory expansion cards, etc.). The first two digits of the version number tell you in which volume and issue of *HCM* the program *initially* appeared. The third digit of the version number indicates the version of the program. When a program initially appears in *HCM*, it is version 1. Any subsequent revisions to the program, if later published in the magazine or in the software available on magnetic media from *HCM*, will bear a revised version number.



PROGRAM LISTING

Your Guide to Typing in Programs from *HCM*

Within these pages is a software bonanza: entertainment, education, home and business applications, utilities, and tutorials—just for you. All you need to do is type them into your computer. *HCM* has taken most of the strain out of this process:

- Typeset listings with numbers in boldface.
- A bold, double vertical bar separating the line numbers from the program statements in BASIC listings.
- A vertical background grid to aid entry of the spaces.
- An error-reducing Bug-Out Code

Looking at the Character Reference Chart (Figure 1, below) will show you how each character actually appears in the program listings. By checking any questionable characters with this chart, you can reduce typing errors to a minimum.

Figure 1: Character Reference Chart



Before You Begin

Since *HCM* publishes for several different computers, the first thing you should do is make sure that you are looking at the listing designed for your machine. If, for example, you have an Apple *II*e, make sure you look for the following black bar above the listing: APPLE II Family The computer model name will likewise appear on each subsequent page of each listing, so always look for the name before you begin typing from a new page of listings.

Before you begin typing-in the program, you will want to set up a system to save your program. Whether you are using a cassette or diskette storage system, now is the time to be certain it is properly connected, powered up, and loaded with a blank cassette or an initialized disk. As you type-in your program, you should get in the habit of saving your work after every 20 or so lines.

One of the most common errors in entering a listing is typing one symbol for another. These transpositions include substituting the letter O for the number 0, the letter l for the number 1, the letter S for the symbol $, and the uppercase B for the number 8. The last error is especially likely when working in hexadecimal numbers which are composed of the numbers 0-9 and the uppercase letters A-F.

The listings in *HCM* are always the same number of characters wide, but the number of characters put on any line of the video display will vary from computer to computer. Don't try to make your listings *look like* the type-set listing—instead make sure you type the listings character for character and space for space.

The Bug-Out Code (BOC)

All *HCM* programs are listed with Bug-Out Codes (BOC—pronounced bee-oh-see) that are found in the first column of each program listing. The BOC is represented by an uppercase letter and appears before every line number. Take note: this character is **not** to be typed-in. One of the main reasons that we are using letters instead of numbers is to make sure that the BOC is not entered as part of a line number.



To use BOCs for keeping bugs from creeping into your typed-in programs, see pages 77-79.

A Special Note to TI Users:

Due to architecture and language constraints with the TI-99/4A, users must have both Extended BASIC and a disk drive in order to use the BOC. We are currently developing a BOC program that will work with the Editor/Assembler Cartridge, and hope to have it soon.

A Special Note on Atari Listings

Atari has more special graphic characters available from the keyboard than any other computer that we cover. Reproducing these characters can often take up to three different key strokes. Instead of forcing our readers to translate each of these special characters into the correct key press, we include key-stroke intructions that are placed between two hands with pointing fingers. For example, when you find ☞CTRL ,☜ in an *HCM* listing, you will know to hold down the [CTRL] key and press the [,] key. The result of this key press is the ♥ symbol.

These instructions will include a number if the operation is to be repeated. For example, ☞5 CTRL A☜ tells you to hold down the [CTRL] key and press the [A] key 5 times.

Whenever you see the [CTRL] or [SHIFT] keys between two hands, hold them down while you press the key that follows them. The [ESC] key is an exception. This key represents a separate key press. So, when you see a ☞ESC CTRL +☜ you are to first *tap* the [ESC] key, and *then* hold down [CTRL] and press the [+] key.

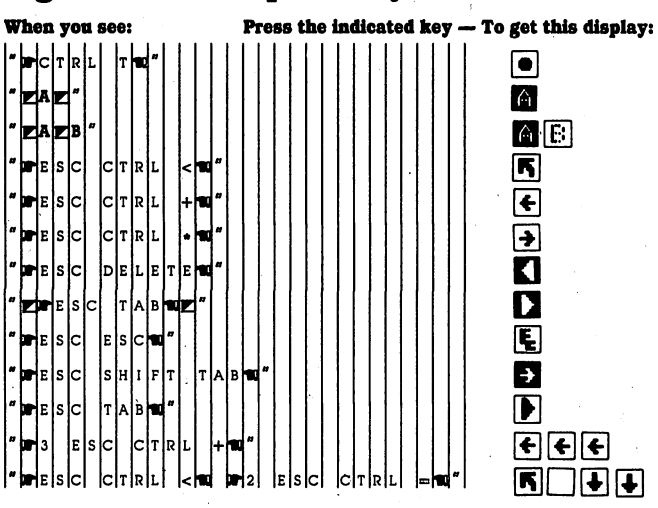
The Atari has one special key that none of the other computers have. This key looks like this ◪ and is used to print reverse characters. Whenever you come across ◪ in a listing, simply press this key.

When you come to the hand symbols, remember:

- **Each operation is enclosed in its own set of hand symbols.**
- **If any key action requires you to press [CTRL] or [SHIFT], press the control key or the shift key *first* and hold it down before pressing the second key.**
- **Both the [ESC] and ◪ keys are single key presses and should be pressed independently of the keys that follow them.**
- **Everything between a pair of hand symbols is set in a different typeface.**

In Figure 2 below, we have included a chart showing you a representative sample of the symbols that appear when you use keystrokes enclosed by the hand symbols.

Figure 2: Atari Special Symbols



A Special Note on C-64 Listings

Commodore uses more than 90 special symbols to represent various keyboard operations: for instance, the symbol [♥] in a program represents the operation of holding down the [SHIFT] key and pressing the key which has CLR on its upper half (second key from the right on the top row). This operation clears the screen.

Rather than reproducing these symbols, *HCM's* listings include key-stroke instructions, between two hands with pointing fingers. For example, when you find SHIFT CLR in an *HCM* listing, you will know to hold down the [SHIFT] key and press the key with CLR on it.

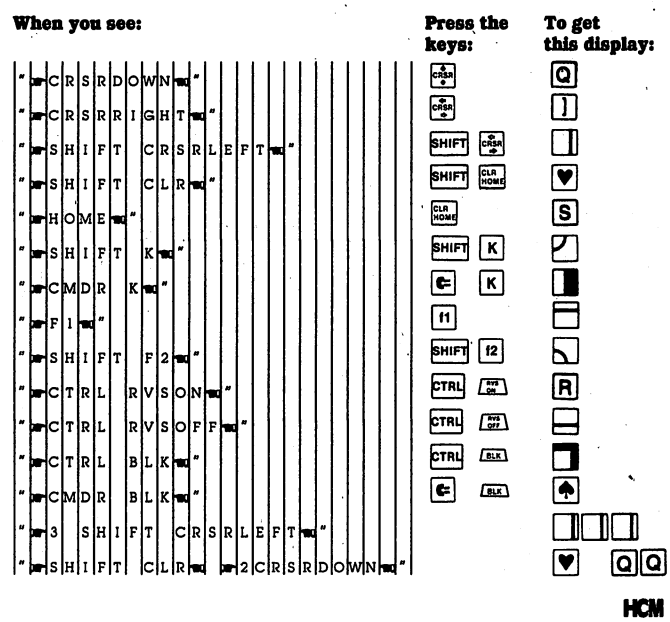
A number is included if you need to repeat the operation: 8 SHIFT CRSRLEFT tells you to hold the [SHIFT] key down and press the cursor left key (on the bottom right of the keyboard) 8 times.

When you come to the hand symbols, remember:

- **Each operation is enclosed in its own set of hand symbols.**
- **If any key action requires you to press two keys, press the control key, the Commodore key, or the shift key *first* and hold it down before pressing the second key.**
- **Everything between a pair of hand symbols is set in a different typeface.**

In Figure 3, we have included a chart showing you a representative sample of the symbols that appear when you use keystrokes enclosed by the hand symbols. (Notice that the hand symbols always appear within quotation marks—as in a print statement.)

Figure 3: C-64 Special Symbols





HCM BUG-OUT

Bug-Out is an error detection program for catching type-in mistakes. It is available for every computer brand *HCM* covers—Apple, Atari, Commodore, IBM, and Texas Instruments. When you use this utility, typos are easily found and corrected.

Before you type-in another *HCM* program, type-in the *Bug-Out* program specified for your computer. Because a properly typed-in *Bug-Out* routine is essential for it to accurately detect typing errors in other programs, be extra careful to ensure accuracy. Once you have it entered, save *Bug-Out* to tape (an option for Atari and Commodore only) or disk.

Comparing Two Sets of BOC's

When you look at our listings, you will notice an upper-case letter placed in the left-most column at the beginning of each program line. Separated from the line numbers by a bold vertical bar, this letter is the correct BOC. Do **not** type these letters in. The BOC is a quality-control character. Each program line is carefully dissected, and mathematically compacted into a single-character representation.

These letters will help you detect key-in errors *after* you have a listing fully entered. When you RUN the *Bug-Out* program as directed below, it will generate another series of BOCs either on the screen, or to a printer. Compare the codes generated by the *Bug-Out* program with the codes published in the left-most column of our listings. If a published BOC for a line is different from a BOC for your typed version of the same line, you will know that line contains a typing error.

How To Do It

1. The first step is to type-in the desired program.
2. After typing the program in, SAVE it as usual.
3. Then also SAVE the program as an ASCII text file—this is the format needed for *Bug-Out* to do its job. Always use a different file name to distinguish between the program file SAVEd in step 2 and this text file. We suggest you add a suffix like .T (or _T on the 99/4A) to the end of the text file name for added clarity. The process of saving programs as ASCII text files on each machine is detailed on page 78 (see "Turning Programs Into Text Files").
4. After you've SAVEd your program as an ASCII text file, make sure that the disk or tape containing the text file is inserted, then RUN the *Bug-Out* program. Once RUN, *Bug-Out* will ask for the name of the ASCII file and whether you want the program's output to go to the screen or the printer. After all this has been entered, the computer will print out its list of BOCs and the corresponding line number. For example:

```
N 100
S 110
Q 120 . . .
```

5. Carefully go through the program listing in the magazine to find, and take note of, all the published BOCs that are different from the BOCs generated by the *Bug-Out* program. Every line that you find with a different BOC code has been typed incorrectly, and should be carefully examined and corrected.

To correct your mistakes (if any), LOAD (OLD on the 99/4A) the program version (not the text file) that you keyed in previously. Now, make the necessary changes to the incorrect program lines and repeat the previous 5 steps until all the BOC codes match. Once they all match, your program should be error free.

REM statements that are not typed correctly will result in erroneous BOCs. If the only differences between a typed-in program and the magazine listing are in REM statements, the program will still RUN as intended. So you needn't waste time concerning every REM statement before running your *HCM* programs. ***Continued***

TURNING PROGRAMS INTO TEXT FILES

Apple II Family: The Apple method of making a program into a text file requires merging your typed-in program with a short *Capture* program (see below). [Note: in *HCM* Vol. 5, No. 5, the *Capture* program was regretably omitted.—Ed.] Our version is based on the *Capture* program found on page 140 of the *BASIC Programming with ProDOS* manual published by Apple. If you do not have either *Apple Programmer's Assistant (APA)* or *Renumber* program in DOS 3.3, you must type-in *Capture* at the same time as the program you are **SAVE**ing as a text file.

In either case, with *Capture* (at lines 1 through 10), and the program you wish to capture as a text file in memory, just type **RUN**. The *Capture* program then **LIST**s the program in memory to disk as a text file under whatever name is **INPUT** when *Capture* first starts running. Because *HCM* programs always begin with line 100, the *Capture* program always **LIST**s starting at line 100, and does ***not*** **LIST** lines 1-10, which do the actual capturing.

Atari disk users enter: LIST "D:PRGNAME.**T"**
Atari tape users enter: LIST "C:PRGNAME.**T"**

C-64 disk users enter: OPEN 8,8,8,"PRGNAME.**T,S,W" : CMD 8 : LIST**
When the cursor returns, enter: **PRINT#8 : CLOSE 8**
C-64 tape users enter: OPEN 1,1,1,"PRGNAME.**T" : CMD 1 : LIST**
When the cursor returns, enter: **PRINT#1 : CLOSE 1**

IBM PC and PCjr users: SAVE the program with the ASCII option like this: **SAVE"**PRGNAME.**T",A**

TI-99/4A users: You can **LIST** your program to disk from Extended BASIC by typing the following, with your newly typed-in program residing in memory: **LIST "DSK1.**PRGNAME**_T"**

Special Note to Apple Users:

Applesoft BASIC has an idiosyncracy concerning **REM** and **DATA** statements that our Bug-Out program takes into consideration. To ensure identical BOCs, always make sure your **REM** and **DATA** statements are typed *exactly* as they appear in the listing. Although **REM** statements do not affect a program's performance, **DATA** statements are often a source of typing bugs. Therefore, after either typing in or editing a **DATA** statement, be sure that it has the same number of spaces as in the magazine.

CAPTURE

APPLE II Family

```
10  D$ = CHR$ (4): PRINT D$"OPEN HOME : PRINT "LIST TO WHAT FILE NAME (
20  PRINT "LIST NAME PROGRAM FILE NAM
    E";: INPUT FL$: PRINT D$"OPEN"FL$",D1"
30  PRINT D$"CLOSE": PRINT D$"DELETE"FL$",D1"
40  PRINT D$"OPEN"FL$",D1"
50  PRINT D$"WRITE"FL$: LIST 100-: PRINT D$"CLOSE"FL$
60  PRINT "***DONE***": END
```

BUG-OUT

APPLE II Family

```
100  REM  ***********
110  REM  *  BUG-OUT  *
120  REM  ***********
130  REM  COPYRIGHT 1985
140  REM  EMERALD VALLEY PUBLISHING CO.
150  REM  BY THE HCM STAFF
160  REM  HOME COMPUTER MAGAZINE
170  REM  VERSION 5.1
180  REM  APPLE II FAMILY APPLESOFT
190  D$ = CHR$ (4): CR$ = CHR$ (13): SP$ = CHR$ (32)
200  TEXT : HOME : PRINT TAB( 10);"HCM BUG-OUT PROGRAM": PRINT : PRINT "THIS PROGRAM, LISTED TO DISK, WILL CONVERT ANY": PRINT "PROGRAM TO A TEXT FILE.": PRINT : PRINT "INVERSE CAPITAL LETTERS": PRINT "WILL BE PRINTED WHEN THIS": PRINT "PROGRAM IS RUN.": PRINT : PRINT "PLACE DISK WITH": PRINT "PROGRAM TO BE": PRINT "CONVERTED IN DRIVE": PRINT : INPUT "1, 2 OR 3 ? "; DR$: IF DR$ = "" THEN DR$ = "1"
210  VTAB 8: HTAB 1: PRINT "PLACE DISK WITH": PRINT "PROGRAM TO BE": PRINT "CONVERTED IN DRIVE": PRINT : INPUT "1, 2 OR 3 ? "; DR$: IF DR$ = "" THEN DR$ = "1"
220  ONERR GOTO 380
230  PRINT D$;"VERIFY "; FL$;",D1"
240  PRINT D$;"OPEN "; FL$;",D1"
250  VTAB 11: HTAB 1: PRINT "OUTPUT TO": PRINT : PRINT "1. SCREEN": PRINT "2. PRINTER": PRINT : INPUT "1 OR 2 ? "; SLOT
260  GET K$: IF K$ < "1" AND K$ > "2" THEN 260
270  PRINT K$: IF K$ = "1" THEN 290
280  PRINT D$;"PR#1": PRINT D$;"READ "; FL$
290  GET C$: IF C$ = CR$ THEN 300: IF C$ = SP$ THEN 340
300  IF C$ = "" THEN 290: GOTO 320
310  IF C$ = CR$ + "" THEN C$ = CR$: GOTO 320
320  A$ = A$ + C$: GOTO 300
330  IF A$ = "" THEN 300
340  IF LEFT$ (A$,1) = SP$ THEN A$ = RIGHT$ (A$, LEN (A$) - 1): GOTO 340
350  IF VAL (A$) = 0 THEN PRINT A$;: GOTO 370
360  PRINT VAL (A$);: B$ = STR$ (VAL (A$)): GOSUB 500
370  PRINT SP$;: GOTO 340
380  POKE 216,0: IF PEEK (218) + PEEK (219) * 256 > 43008 THEN POKE 218,255: POKE 219,175
390  PRINT D$;"OPEN"FL$: PRINT D$;"DELETE"FL$: PRINT D$;"OPEN"FL$: PRINT "WITH ERRORS": GOTO 440
400  PRINT D$;"CLOSE": PRINT D$;"PR#0": END
410  PRINT "IS NOT ON DISK": GOTO 440
420  PRINT "I/O ERROR—CHECK DRIVE": GOTO 440
430  PRINT "ILLEGAL FILE NAME—TRY AGAIN": GOTO 440
440  CALL -3288: PRINT "PRESS ANY KEY": GET K$: RUN
450  PRINT "DETECTED AT LINE": PRINT "STARTING AGAIN": GOTO 440
460  END
470  C$ = "": FOR I = 1 TO LEN (A$): C1 = ASC ( MID$ (A$, I, 1)): IF C1 = 34 THEN C$ = C$ + CHR$ (34) + CHR$ (34): GOTO 490
480  C$ = C$ + CHR$ (C1)
490  NEXT I: A$ = C$: RETURN
500  IF B$ = "" THEN B$ = "0"
510  IF INT (B$) = B$ THEN B$ = MID$ (B$, 2): GOTO 510
520  RETURN
```

HCM

BUG-OUT

ATARI 800/800XL/130XE

```
100  REM  ***********
110  REM  *  BUG-OUT  *
120  REM  ***********
130  REM  COPYRIGHT 1985
140  REM  EMERALD VALLEY PUBLISHING CO.
150  REM  BY THE HCM STAFF
160  REM  HOME COMPUTER MAGAZINE
170  REM  VERSION 5.5 FOR THE 800, 800XL,
180  REM  ATARI BASIC
     130XE
190  DIM FILE$(16), INS(255), BL$(7): BL$="
     ": POKE 82,0
200  ? "ESC CTRL <▼▲": ? "HCM BUG-OUT": ? "
     ": ? "Program must be saved as a LISTed file**": ? : INPUT "DEVICE & FILENAME";FILE$: IF FILE$="" THEN 330
210  ? "ENTER DEVICE:FILENAME": INPUT FILE$: IF FILE$="" THEN 330
220  TRAP 340: OPEN #1,4,0,FILE$: ? "LIST TO SCREEN": ? "OR PRINTER (S/P)": INPUT K$: IF K$="P" THEN LPRINT "BUG-OUT": LPRINT "HCM": LPRINT "LISTed file**": LPRINT FILE$
230  OPEN #1,4,0,"K:": POKE 764,255: POKE 752,1: ? "PRESS ANY KEY TO BEGIN": IF PEEK(764)=255 THEN 230
240  POKE 764,255: ? "LISTING...": ? : ? : ? : TRAP 40000
250  ? "?  :?  "READING...": ? : ? : ? : TRAP 330
260  INPUT #1;IN$: P=1
270  IF IN$(P,P)<>CHR$(34) THEN P=P+1: GOTO 270
280  CK=0: CK1=0: FOR I=1 TO LEN(IN$): CK1=CK1+ASC(IN$(I,I)): CK=CK+ASC(IN$(I,I))*I: NEXT I: IF CK1<>INT(CK/26)*26 THEN ? "ERROR IN LINE ";IN$(1,P-1): GOTO 260
290  IF IN$(1,6)="DATA " THEN 310
300  IF IN$(1,6)="REM " THEN ? CHR$(CK+65);BL$(1,6-P);IN$(1,P-1): GOTO 260
310  ? CHR$(CK+65);BL$(1,6-P);IN$(1,P-1): GOTO 260
320  ? CHR$(CK+65);BL$(1,6-P);IN$(1,P-1): GOTO 260
330  CLOSE #1: ? : ? "DONE": POKE 764,255: END
340  ? : ? "**IO ERROR**": SOUND 1,50,10,8: FOR I=1 TO 1000: NEXT I: SOUND 1,0,0,0: CLOSE #1
350  GOTO 200
```

HCM

BUG-OUT — COMMODORE 64

```
N 100 REM ***********
Z 110 REM * BUG-OUT *
F 120 REM ***********
M 130 REM COPYRIGHT 1985
R 140 REM EMERALD VALLEY PUBLISHING CO.
O 150 REM BY THE HCM STAFF
N 160 REM HOME COMPUTER MAGAZINE
V 170 REM VERSION 5.5.1
G 180 REM COMMODORE 64 BASIC
B 190 PRINT "[SHIFT CLR][CTRL RVSON]     HCM BUG-OUT"
F 200 PRINT "       PROGRAM MUST HAVE BEEN SAVED"
L 210 PRINT "       AS A SEQUENTIAL TEXT FILE."
Z 220 F$="":INPUT "NAME OF LISTED FILE";F$:IF F$="" THEN END
Y 230 PRINT:PRINT "TAPE OR DISK (T/D)?"
I 240 GET K$:IF K$<>"T" AND K$<>"D" THEN240
Q 250 IF K$="D" THEN270
Q 260 OPEN 1,1,0,F$:GOTO290
P 270 OPEN 15,8,15:OPEN 1,8,2,F$+",S,R":INPUT#15,E,E$
U 280 IF E THEN PRINT "** "E$" **":FOR I=1 TO 1000:NEXT:CLOSE1:CLOSE15:GOTO190
X 290 PRINT:PRINT"LIST TO SCREEN OR PRINTER (S/P)?"
N 300 GET K$:IF K$<>"S" AND K$<>"P" THEN300
E 310 PRINT:PRINT "READING...":IF K$="P" THEN OPEN 4,4
H 320 GET#1,CH$:IF CH$<"0" OR CH$>"9" THEN320
P 330 IN$=CH$
P 340 GET#1,CH$:IN$=IN$+CH$
X 350 IF CH$=CHR$(10) THEN CLOSE1:CLOSE4:CLOSE15:PRINT:PRINT"DONE":END
N 360 IF CH$<>CHR$(13) THEN340
Y 370 P=1
F 380 IF MID$(IN$,P,1)<>" " THEN P=P+1:GOTO380
C 390 CK=0:C1=0:FOR I=1 TO LEN(IN$)-1:C1=C1+ASC(MID$(IN$,I,1))
A 400 IF I/2=INT(I/2) THEN CK=CK-ASC(MID$(IN$,I,1)):GOTO420
D 410 CK=CK+ASC(MID$(IN$,I,1))
J 420 NEXT
P 430 CK=ABS(CK)*C1:I=INT(CK/26):CK=CK-I*26
Q 440 IFK$="S"THENPRINTCHR$(CK+65)LEFT$("      ",6-P)LEFT$(IN$,P),:IN$="":GOTO340
B 450 PRINT#4,CHR$(CK+65)LEFT$("      ",6-P)LEFT$(IN$,P):IN$="":GOTO340
```

HCM

BUG-OUT — IBM PC/PCjr, TANDY 1000

```
I 100 ' ***********
W 110 ' * BUG-OUT *
Q 120 ' ***********
X 130 ' COPYRIGHT 1985
W 140 ' EMERALD VALLEY PUBLISHING CO
H 150 ' BY THE HCM STAFF
U 160 ' HOME COMPUTER MAGAZINE
E 170 ' VERSION 5.5.1
X 180 ' DOS 2.1 AND EITHER
D 190 ' IBM PC WITH BASICA
U 200 ' OR IBM PCjr WITH CARTRIDGE BASIC
H 210 ' TANDY 1000 WITH GW BASIC
F 220 CLS:SCREEN 0:WIDTH 40: SP$=CHR$(32):CR$=CHR$(13):LOCATE 3,10:PRINT "HCM BUG-OUT PROGRAM"
Q 230 PRINT "FILE MUST BE A BASIC PROGRAM SAVED WITH THE A(SCII) OPTION"
K 240 LOCATE 6,3:PRINT "Input File Name: ";:INPUT FL$:IF FL$="" THEN 220 ELSE FL$=LEFT$(FL$,12)
N 250 LOCATE 7,3:PRINT "Which disk drive? A";
I 260 LOCATE 7,21,1:DR$=INKEY$:IF DR$="" THEN 260 ELSE IF INSTR("AaBb"+CR$,DR$) THEN IF DR$=CR$ THEN DR$="A" ELSE 270 ELSE 260
A 270 PRINT DR$:LOCATE 9,3,0:PRINT "Output to (1) Screen":LOCATE 10,13:PRINT "(2) Line Printer"
F 280 LOCATE 12,3:PRINT "Your Choice: "
N 290 LOCATE 12,16,1:DEV$=INKEY$:IF DEV$<>"1" AND DEV$<>"2" THEN 290 ELSE PRINT DEV$:IF DEV$="1" THEN DEV$="SCRN:" ELSE DEV$="LPT1:"
I 300 OPEN DR$+":"+FL$ FOR INPUT AS #1
O 310 OPEN DEV$ FOR OUTPUT AS #2
T 320 IF EOF(1) THEN 350
N 330 LINE INPUT #1,C$ :GOSUB 370
D 340 N=INSTR(C$,SP$):PRINT #2,LEFT$(C$,N);CHR$(65+CK):GOTO 320
I 350 PRINT "THAT'S ALL":END
V 360 ' CHECK SUM
E 370 CK=0:CK1=0:FOR I=1 TO LEN(C$):CHK=ASC(MID$(C$,I,1)):CK1=CK1+CHK:IF I/2=INT(I/2) THEN CK=CK-CHK ELSE CK=CK+CHK
N 380 NEXT I
R 390 CK=ABS(CK)*CK1:I=INT(CK/26):CK=CK-I*26
G 400 OT$=CHR$(65+CK):RETURN
```

HCM

BUG-OUT — TI-99/4A

```
P 100 REM ***********
Q 110 REM * BUG-OUT *
P 120 REM ***********
M 130 REM COPYRIGHT 1985
H 140 REM EMERALD VALLEY PUBLISHING CO
A 150 REM BY THE HCM STAFF
R 160 REM HOME COMPUTER MAGAZINE
G 170 REM VERSION 5.5.1
T 180 REM TI EXTENDED BASIC ONLY
D 190 SP$=CHR$(32):: CR$=CHR$(13)
X 200 DISPLAY AT(2,5)ERASE ALL:"HCM BUG-OUT PROGRAM"
N 210 DISPLAY AT(4,1):"YOU MUST ""LIST"" THE PROGRAM TO DISK." :: DISPLAY AT(6,1):"THEN RUN ""BUG-OUT"""
P 220 DISPLAY AT(8,1):"FILE NAME: DSK1." :: ACCEPT AT(8,12)SIZE(-15):FL$
F 230 ON ERROR 520
R 240 OPEN #2:FL$,INPUT ,DISPLAY ,VARIABLE 80
Q 250 DISPLAY AT(10,1):"SEND OUTPUT TO: 1. SCREEN" :: DISPLAY AT(11,17):"2. PRINTER"
M 260 DISPLAY AT(12,1):"1" :: ACCEPT AT(12,1)VALIDATE("12")SIZE(-1):DV :: DV=DV-1
C 270 IF DV=0 THEN 310
T 280 ON ERROR 530
T 290 DISPLAY AT(13,1):"PRINTER:" :: DISPLAY AT(14,1):"RS232" :: ACCEPT AT(14,1)SIZE(-28):DV$
W 300 OPEN #1:DV$,OUTPUT
N 310 ON ERROR STOP
Z 320 IF EOF(2)THEN 500
A 330 LINPUT #2:C$
E 340 IF C$="" THEN 320
Y 350 N=POS(C$,SP$,1):: OT$=SEG$(C$,1,N):: LNNUM=VAL(OT$):: IF LEN(C$)<80 THEN 400
A 360 FLAG=1 :: IF EOF(2)THEN FLAG=0 :: GOTO 400 ELSE LINPUT #2:D$
Z 370 M=POS(D$,SP$,1):: IF M>5 THEN 390
Y 380 ON ERROR 540 :: LN=VAL(SEG$(D$,1,M)):: IF LN>LNNUM THEN 400
Y 390 C$=C$&D$ :: FLAG=0
M 400 CK$=C$ :: M1=POS(C$," REM ",N):: M2=POS(C$," ! ",N):: IF M1<1 AND M2<1 THEN 450
A 410 IF M1=N THEN CK$=SEG$(C$,1,N+4):: GOTO 450
A 420 IF M2=N THEN CK$=SEG$(C$,1,N+2):: GOTO 450
H 430 IF M1>N THEN CK$=SEG$(C$,1,POS(C$," REM ",1)+4):: GOTO 450
Y 440 IF M2>N THEN CK$=SEG$(C$,1,POS(C$," ! ",1)+2)
W 450 CK=0 :: CK1=0 :: FOR I=1 TO LEN(CK$):: CHK=ASC(SEG$(CK$,I,1)):: CK1=CK1+CHK :: IF I/2=INT(I/2)THEN CK=CK-CHK ELSE CK=CK+CHK
M 460 NEXT I
C 470 CK=ABS(CK)*CK1 :: I=INT(CK/26):: CK=CK-I*26
W 480 OT$=OT$&SP$&CHR$(65+CK)
S 490 PRINT #DV:OT$ :: IF FLAG=0 THEN C$="" :: GOTO 320 ELSE C$=D$ :: FLAG=0 :: GOTO 350
A 500 CLOSE #2 :: PRINT "ALL DONE" :: IF DV=1 THEN CLOSE #1
H 510 END
H 520 DISPLAY AT(10,1)BEEP:"FILE NOT FOUND" :: GOSUB 550 :: GOTO 200
P 530 DISPLAY AT(14,1):"PRINTER NOT RECOGNIZED" :: GOSUB 550 :: RETURN 280
X 540 CALL ERR(EN,ET,ES,EL):: IF EN=74 THEN RETURN 390 ELSE PRINT "ERROR NUMBER: ";EN:" IN LINE ";EL :: STOP
M 550 FOR DE=1 TO 500 :: NEXT DE :: RETURN
```

HCM

Card-Shuffler

REMARKS

The most important routine in the TI version of *Card-Shuffler* is the Get-A-Folder routine, located in lines 340-480. This routine searches various folders and creates a record of each *Card-Trix* card that meets the search parameters' specifications. Without this routine, the program is useless.

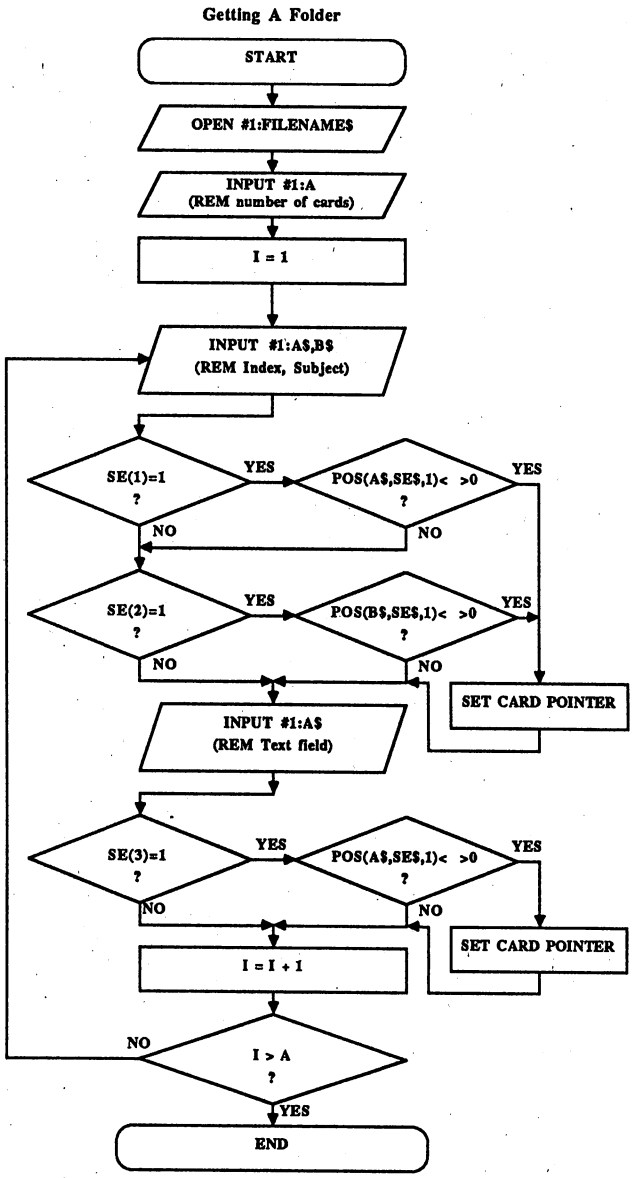
When the Get-A-Folder routine finds a card, it does not store the actual card in memory—it stores a pointer to the card. As a result, memory can hold as many as 4 card folders at one time. The C(,) array contains the card pointers.

We use search parameters to specify what the routine is to search for. The search parameters consist of the item being searched for and the fields to be searched through. The SE$ variable contains the item being searched for. The three elements of the SE( ) array act as flags indicating which, if any, of the three fields should be searched. If SE(1) equals one, the routine searches through the first field: the Index field. If SE(2) equals one, the routine searches through the second field, the Subject field. Similarly, SE(3) controls the third field, the Text field.

The Design Focus on this page presents a flow chart of this routine. First, the routine OPENs the desired file and INPUTs the first item. The first item of a *Card-Trix* file informs us of the number of cards that are in the folder. The variable A stores this number. Next, we INPUT the first two fields of a card—the Index and Subject fields. Using the SE( ) array, the program checks whether it should search through these first two fields. If it should, the program searches through each field using the POS( ) function. When the POS( ) function returns a non-zero value, the routine sets a pointer to the current card. Next, the routine INPUTs the Text field. The program checks SE(3) to see if this field should be searched. The program uses the POS( ) function to search the Text field, as well. Once again, if the POS( ) function returns a non-zero value, the routine sets a pointer to the current card. Using a FOR-NEXT loop, this process repeats the number of times specified by the variable A.

HCM Glossary terms: array, element, field, function, parameter.

DESIGN FOCUS



LISTING ANNOTATIONS

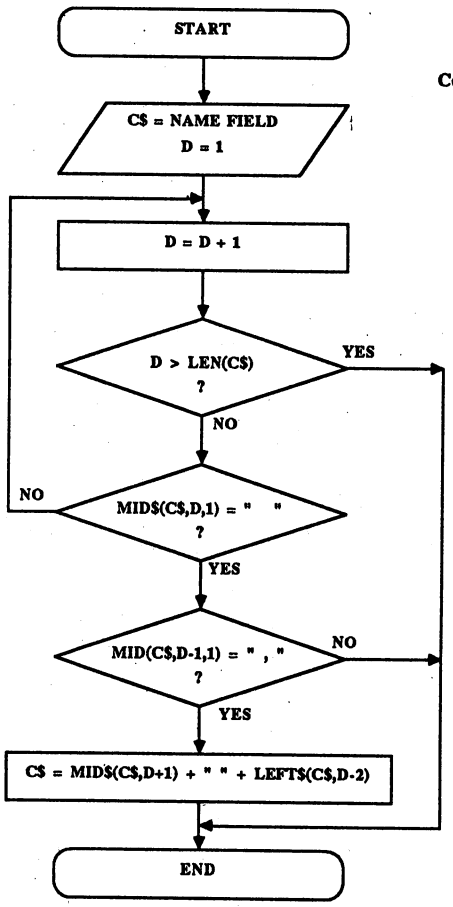
Line Nos.	
100-180	Program header
190-200	Initialize program
210	Title screen
220-280	Main menu
290-330	Search cards menu
340-480	Get a folder
490-570	Toss a folder
580-660	Set search parameters
670-860	Print folder index
870-1090	Save indexed folder
1100-1120	Exit program
1130	Erase bottom of screen
1140-1170	Display search parameters
1180-1190	Draw a box
1200-1260	Input file name
1270	Print "WORKING"
1280	Error routine
1290	Character data

DIRECTORY OF VARIABLES

Variables	Functions
A	General utility
B	General utility
C(,)	Pointers to the cards that are found
C$(,)	Data for one card
FN$( )	Folders' file names
FS$( )	Folders' search item
I	Loop counter
J	Loop counter
K	ASCII of keyboard input
MX	Maximum number of cards
MXF	Maximum number of folders
NF	Current number of folders
PR	Flag for Printer/Screen output
S	Keyboard status
S$	General utility
SE$	Item searched for
SE( )	Fields searched through

Electronic Postman

DESIGN FOCUS



Convert Name Field Into "First Last" Format

LISTING ANNOTATIONS

Line Nos.	
100-190	Program header
200-260	Display title screen
270-320	Initialize program
330-350	Main loop
360-520	General menu routine
530-760	Edit file
770-860	Sort file
870-1000	Search file
1010-1120	Item is found
1130-1460	I/O routines
1130-1160	Ask user if load or save
1170-1190	Input file name
1200-1240	Input device (tape/disk)
1250	Jump to load or save routine
1260-1360	Load file
1370-1460	Save file
1470-1690	Print routines
1700-1970	Merge routine
1980-2090	Start the merging process
2100-2200	Find merge code
2210-2280	Code has been found
2290-2340	Put name in "First Last" format
2350-2440	Print one line of merged text
2450-2590	Load *Electronic Typewriter* text
2600-2640	Load template
2650-2740	Update editing screen
2750-3260	Draw editing screen
3270-3290	Read disk error
3300-3330	Buzz and pause
3340-3400	Exit program
3410-3420	Plot cursor at X,Y
3430-3480	Menu data
3490-3870	Machine language data for input routine

REMARKS

The Electric Merge in the Commodore 64 version of *Electronic Postman* has a very useful feature: It enables you to specify a person's first or last name. If you use a merge code of *F, for example, you are instructing the computer to replace the code with the first name only. Conversely, *L specifies the last name.

Electronic Postman uses a short subroutine located in lines 2290-2340 to separate the first and last name of a Name field entry. Note that there are two different formats in which you can enter a name: You can enter a name as "Last, First" or "First Last." Take the name John Smith, for example. With *Electronic Postman* you have the option of entering this name as "Smith, John" or, as one might normally write it, "John Smith."

The Design Focus on this page provides a flow chart of the name-separating subroutine. First, this subroutine sets the variable C$ equal to the Name field of the current record and initializes D, our loop counter, to equal one. Using D as an index, *Electronic Postman* searches for the first space in C$. If it finds a space, the program checks to see if a comma precedes that space. If so, the program eliminates the comma from C$ and switches the first and last names. If it does not find a comma, the subroutine ends because C$ is already in "First Last" format. When the subroutine ends, C$ is in "First Last" format with the variable D equal to the location of the separating space.

HCM Glossary terms: field, subroutine.

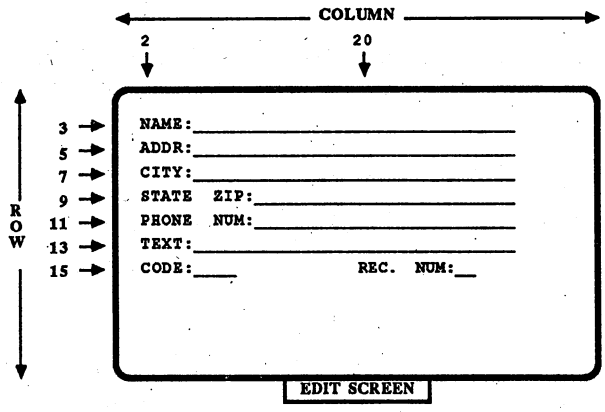
DIRECTORY OF VARIABLES

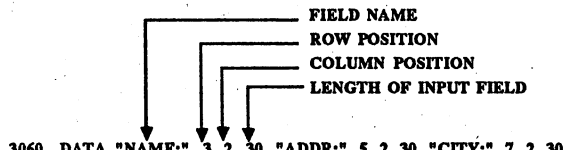
Variables	Functions
A	Utility variable
BL$	80 blank characters
C$	Used in merge routine
CD	Number of merge code (1-8)
CD$	String containing merge codes
D	Utility variable
E	Disk error number
E$	Disk error descriptor
EX	Exit flag
F$	*Electronic Typewriter* (ET) file name
I	Loop counter
J	Loop counter
K	ASCII of keyboard input
K$	Keyboard input character
L	Utility variable
LM	Left margin
LN	Current line of ET text
M	Current menu
M$( )	Menu options
MC	Menu option chosen by user
MD	Current field in edit screen
ME( )	Menu data array
MX	Maximum number of records
N	Current record number
N$(,)	Record data
RM	Right margin
S	Utility variable
S$	Utility string
SC$	Password for merge
SP	Spacing
T$	Program title
TP$( )	Template array
TX$( )	Text array
X	Screen column
Y	Screen row
Z	Utility variable

Electronic Postman

DESIGN FOCUS

DATA Statements Used In Formatting The Edit Screen






```
3060 DATA "NAME:", 3, 2, 30, "ADDR:", 5, 2, 30, "CITY:", 7, 2, 30,
     "STATE ZIP:", 9, 2, 25, "PHONE NUM:", 11, 2, 25, "TEXT", 13,
     2, 30, "CODE:", 15, 2, 4, "REC. NUM:", 15, 20, 2
```

LISTING ANNOTATIONS

Line Nos.	
100-190	Program header
200-280	Program main line
290-330	Main menu
340-450	Edit file
460-640	Sort file
650-770	Search file
780-850	Item is found
860-920	Ask user if load or save
930-1020	Load file
1030-1080	Save file
1090-1240	Print file
1250-1460	Electric merge
1470-1590	Merging and printing
1600-1710	Find "*" code
1720-1800	Merge code has been found
1810-1860	Put name in "First Last" format
1870-2000	Print one line of *Electronic Typewriter* text
2010-2080	Exit program
2090-2130	Edit screen
2140-2260	Edit an item on edit screen
2270-2290	Enter a number
2300-2410	Enter a line with cursor control
2420-2670	Enter file name
2680-2690	Turn on printer
2700-2710	Turn off printer
2720-2760	Configure menu
2770-2850	Operate menu
2860-2920	Ring bell twice and pause
2930-3070	Program initialization
3050	Keypress data
3060	Editing screen data
3080-3270	Error handler
3080-3110	Read error and branch accordingly
3120-3270	Print error

REMARKS

The Apple version of *Electronic Postman* uses a very structured method to format the program's edit screen. The DATA statement in line 3060 contains all the formatting information that the program needs to arrange the edit screen's 8 fields. This data is in the following format: 3060 *field name, screen row position, screen column position, length of field . . .*

We use this data to get the different *field names* and place them on the screen at the indicated *row* and *column* position. When receiving input for one of these fields, the program uses the *length of field* data to determine the maximum number of characters that can be entered. See the Design Focus for an illustration of how line 3060 controls the edit screen's format.

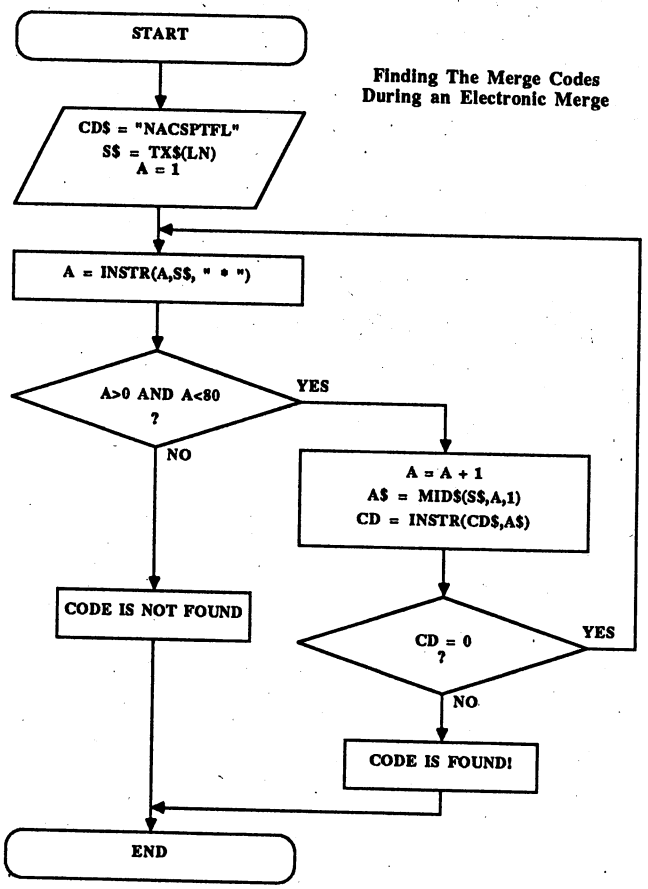
HCM Glossary terms: field.

DIRECTORY OF VARIABLES

Variables	Functions
A,B,D,N,X,Y	General purpose
B80$	String of 80 spaces
BK$	One space
BL$	CHR$(7), bell sound
BX$	CHR$(2), control B
C	*Electronic Typewriter* (ET) line code
C$,S$,X$	General purpose
CD	Merge code value
CD$	String of merge codes
CR$	CHR$(13), carriage return
D$	CHR$(4), wakes up DOS
DI,IT,JI,KI,SI	Loop counters
DN$	CHR$(10), cursor down
DR$	Number of disk drives
DX$	DELETE command character
E	Error code
EC$(,)	Edit command codes and field names
EC(,)	Edit page field locations and lengths
EN$( )	Keyboard entry array
EOL	Address of "clear to the EOL" routine
ESC$	CHR$(27), ESC character
FL$,FT$	File names
FL$( )	Keyboard entry array for file name
GC	Used in "get char" routine
HB,HC	Horizontal screen locations
HX	Pointer during keyboard entry
IN$	Single character entry
IX$	Insert command character
JP	Field selector on edit screen
LF	Length of field in file-name entry
LF$	CHR$(8), cursor left
LM,RM	Left and right margins in ET file
LN	Line pointer in ET file printout
LS	Length of field on edit screen
MC( ),MN$( )	General menu routine parameters
MD	Number of current field being edited
MN(,)	Screen locations for menu items
MX	Maximum number of records
N$(,)	Records
N1,N2,N3,N4	Number entry parameters
P1,P2	First and last record to print
RS	Controls returns from error routine
RT$	CHR$(21), cursor right
S	Save number of current record
S$( )	Temporary storage for sort routine
SC$	Electronic merge code
SP	Spacing for ET file
SV	Template flag when loading an ET file
SX$	Null string
TM	Maximum number of lines in ET file
TP$( )	Line codes in ET file
TX$( )	Text line in ET file
UP$	CHR$(11), cursor up
VC,VF	Vertical screen locations

Electronic Postman

DESIGN FOCUS



Finding The Merge Codes During an Electronic Merge

LISTING ANNOTATIONS

Line Nos.	
100-210	Program header
220-280	Display title screen
290-360	Initialize program
370-390	Main loop
400-560	General purpose menu routine
570-770	Edit file
780-980	Search file
990-1180	I/O routines
1190-1390	Print file
1400-1490	Sort file
1500-1570	Exit program
1580-1690	Display edit screen
1700-1740	Update editing screen
1750-1870	Display various kinds of boxes
1880-2050	Main section of input routine
2060-2150	Place character on screen and in string
2160-2180	Cursor right
2190-2210	Cursor left
2220-2240	Toggle insert mode
2250-2270	Backspace
2280-2300	Delete a character
2310-2340	Buzz and pause
2350-2370	Trap file error
2380-2420	Display file error
2430-2480	Get file name
2490-2660	Load text file
2670-2840	Merge routine
2850-2960	Start merging and printing
2970-3040	Find "*" code
3050-3120	Code has been found—merge code
3130-3180	Put name in "First Last" format
3190-3290	Print one line of *Electronic Typewriter* text
3300-3360	Menu data

REMARKS

The IBM computer's INSTR function can be very useful, especially in a program like *Electronic Postman*. During an Electric Merge, the computer must search through an entire text file for merge codes. The INSTR makes this task much easier. The Design Focus on this page provides a flow chart of the routine that searches for merge codes (lines 2970-3040).

The Electric Merge must look for 8 different codes—represented by the letters N, A, C, S, P, T, F, and L. To shorten the program, we set the variable CD$ equal to "NACSPTFL". Using the INSTR function, the program searches through a line of text from left to right, looking for asterisks. For each asterisk that the program finds, we assign A$ equal to the character directly following the asterisk. Using the INSTR function again, we determine whether A$ is a proper merge code. The computer accomplishes this with the following code:

```
CD = INSTR(CD$,A$)
```

If CD equals zero, A$ is not a proper code. Otherwise, CD indicates which code has been found.

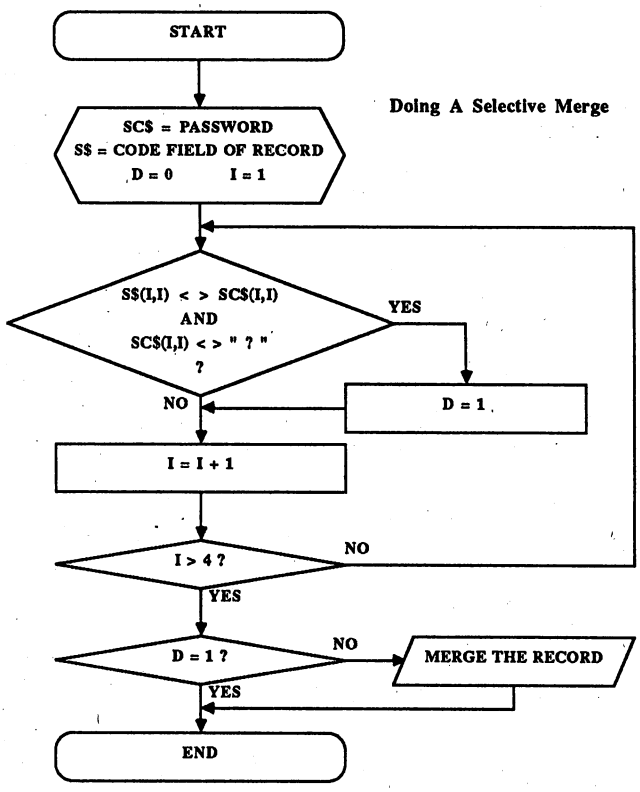
HCM Glossary terms: function.

DIRECTORY OF VARIABLES

Variables	Functions
A	Counter to string position when searching for "*" code
B	Used in search routine
BL$	String of blank characters
CD	Number of merge code (1-8)
CD$	String containing merge codes
CS	Number of characters in input
D	Utility variable
D1	Utility variable
EX	Exit flag
F$	File name
FC$	First character accepted by input
LC$	Last character accepted by input
INS	Insert flag for input routine
K$	String used to input from keyboard
L(,)	Array with edit screen data
LM	Left margin for *Electronic Typewriter* (ET) text
RM	Right margin for ET text
LN	Current line in ET text
M	Current menu option
M$( )	String array with menu options
MC	Menu option chosen by user
MD	Current field in editing field
ME( )	Array with menu data
MX	Maximum number of records
N	Current record number
N$(,)	Record data
P$	Prompt string for file routine
PX	Horizontal position for "Get File Name" function
PY	Vertical position for "Get File Name" function
QT	Flag used to remove quotation marks
S$	String returned in input routine
SC$	Password for Electric Merge
SP	String position for input routine
SPACES	Line spacing for ET text file
T$	Program title
TF$	Previously entered file name
TP$( )	ET template array
TX$	ET text array
UL, UR, LL	Various parts of a box
LR, H1, H2	Various parts of a box
V1, V2	Various parts of a box
X, Y, W, H	Parameters for box drawing
Z	Utility variable

Electronic Postman

DESIGN FOCUS



REMARKS

The Electric Merge in the Atari version of *Electronic Postman* contains a useful Selective Merge option. With this option, you can selectively specify which of your records will merge with your *Electronic Typewriter* text file. The routine located in lines 2180-2220 contains the heart of the Selective Merge option. This routine takes the current Code field of a record and compares it with the password. If there is a match, the program merges the record. If not, the program goes on to the next record.

When comparing the Code field with the password, the program takes wild cards (question marks) into account. If a password contains a question mark, the program ignores the corresponding Code field character. The program uses the variable D to flag discrepancies between the Code field and the password.

HCM Glossary terms: field, record.

LISTING ANNOTATIONS

Line Nos.	
100-190	Program header
200	Set up program
210-250	Display title screen
260-350	Initialize program
360-390	Main loop
400-460	Menu handler
470-560	Edit file
570-730	Bubble sort
740-850	Search routine
860-890	Item is found
900-910	Item is not found
920-930	Call machine language search routines
940-1020	Get file name
1030-1150	Disk I/O
1160-1180	Cassette I/O
1190-1270	Load file
1280-1360	Save file
1370-1490	Load *Electronic Typewriter* (ET) text
1500-1640	Print file
1650-1740	Process formatting code for ET text
1750-1770	Handle ET "B" line code
1780-1820	Handle ET "C" line code
1830-1870	Handle ET "D" line code
1880-1930	Handle ET "E" line code
1940-2010	Print routines for ET text
2020-2040	Set up merge
2050-2170	Process merge
2180-2270	Do selective merge
2280-2410	Find the "*" code
2420-2530	Found code—merge it
2540-2620	Put name in "First Last" format
2630-2650	Print a merged line
2660-2670	Update string index
2680-2750	Update edit screen
2760-2870	Draw edit screen
2880-2910	Exit program
2920-3160	Input routine
3170-3300	CIO routine—I/O handler

DIRECTORY OF VARIABLES

Variables	Functions
A	General purpose
ADD	Used in CIO routine
ADH	Used in CIO routine
ADL	Used in CIO routine
BL$	Eighty blank characters
C$	Used in merge routine
CD	Number of merge code (1-8)
CFLAG	Flag for cassette I/O
CHN	Used in CIO routine
D	General purpose
F$	*Electronic Typewriter* (ET) text file name
FIDX	Field index
FIND$	Used in search routine
FLEN	Field length
FOFF( )	Array of field offsets
I,J, W	Loop counters
INDENT	Indent value for ET line codes
INDENT$	String representation of indent codes
IOCB	Used in CIO routine
IOFLAG	Used in I/O routine
K	ASCII of keyboard input
L	General purpose
LM	Left margin
LN	Current line of ET text
LSPACE	Line spacing for ET text
MAXLEN	Maximum line length of ET text
MC	Menu option chosen by user
MD	Current field in edit screen
ML$	Machine language for flipping between inverse and normal text
ML2$	Machine language used in input routine
N	Current record number
N$	Record data
NMH	Used in CIO routine
NML	Used in CIO routine
NUM	Used in CIO routine
P	Used in formatting ET text
RIDX	Record index
RM	Right margin
RWF	Used in CIO routine
S	General purpose
S$	General purpose
S1$	Used in sort
S2$	Used in sort
TFLAG	Load template?
TP$	ET template string
TX$	ET text string
Z	General purpose

Serf City

REMARKS

Efficient allocation of the workforce is critical to the success of a *Serf City* kingdom, as it is to any economy. The TI version of *Serf City* has a built-in algorithm that allocates workers to various occupations according to a specific hierarchy. The Design Focus illustrates the structure of this algorithm.

First the program automatically deducts soldiers from the workforce. It then tries to fill the fields with sufficient workers to harvest the wheat. To determine the number of people to allocate to field work, the program takes the number of bushels of wheat to be planted, then places one person in the fields for each bushel—if the population is sufficient. If the population is not large enough to plant all the seed, the program places all of the available workers in the fields, however, it reduces the number of bushels planted to equal the number of workers.

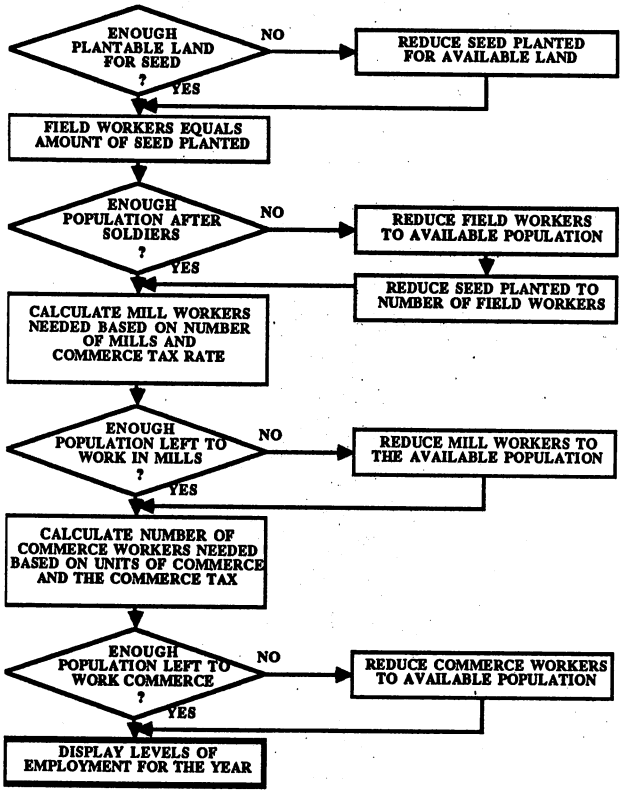
After the program pulls field workers and soldiers from the population, it staffs the mills. The amount of people that the program can allocate to the mills depends on the remaining population, the number of mills, and the commerce tax rate. A high commerce tax rate restricts the number of people that the program can employ in each mill.

After fully staffing the mills, the program employs any remaining population in commerce. The program determines the amount of people that can be employed in commerce according the the remaining population, the number of units of commerce, and the commerce tax rate.

If the kingdom has additional population after the fields, mills, and commerce are fully staffed (in that order), the program considers this remainder unemployed.

DESIGN FOCUS

Serf City Employment



LISTING ANNOTATIONS

Line Nos.	
100-190	Program header.
200-340	Title screen and initialization
350-390	Players' names
400-510	Main control loop
520-620	Main data screen
630-820	Map screen
830-980	Move cursor
990-1250	Add unit of land to map
1260-1460	Resolve conflict
1470-1570	Add mill
1580-1680	Add unit of commerce to map
1690-1770	Update string array S$( )
1780-1850	Wheat option menu
1860-1930	Feed people
1940-2010	Plant seed
2020-2090	Store wheat
2100-2170	Buy wheat
2180-2250	Sell wheat
2260-2380	Input routine
2390-2440	Bad input messages
2450-2730	Hire/fire soldiers
2740-2970	Set tax rate routine
2980-3760	Year-end report routine
3770-3910	End-of-game routine
3920-4020	Print main data screen
4030-4080	Print map screen
4090-4270	Initialize program variables
4280-4290	Map screen format data
4300-4310	Character definition data
4320-4340	Time delay routine
4350-4370	Key scan routine

DIRECTORY OF VARIABLES

Variables	Functions
P$( )	Players' names
S$( )	Map screen
K(,)	Kingdom information
A$	Input routine utility
B$	Input routine utility variable
I1$, I2$	Legal keys list
A	Value returned from input routine
A0, A1, A2, A3	Constants for 0, 1, 2, and 3
A4, A5, A6, A7	Constants for 4, 5, 6, and 7
A8, A9	Constants for 8 and 9
C	Character under cursor
C1	Character above cursor
C2	Character below cursor
C3	Character left of cursor
C4	Character right of cursor
CH	Character placed into map
DN	End-of-game flag
E	Employment in fields
F	Employment in mills
G	Employment in commerce
P	Prosperity of people
P1	Current player
P2	Opposing player
R	Random number
S	Status from scanning keyboard
VW	Value of wheat during year
W	Key returned from key scan
X	Screen coordinates
Y	Screen coordinates
YR	Current year of simulation
Z	Utility loop counter

REMARKS

In the Commodore version of *Serf City*, prosperity plays a major role in population control and tax income. For the purposes of this program, prosperity is a calculation based on the wheat available per person, the ratio of mill workers to population, and the personal tax rate.

The calculated value for prosperity is stored in the variable P. P = wheat for food / (population * 5) * (mill workers + 1) / (population * .1) * ABS ((110 - personal tax rate) * .01).

Because each person requires 5 bushels to survive, the program first multiplies 5 by the number of people, then divides the total amount of wheat allocated to food by this value. This results in the amount of wheat available per person. The next factor that contributes to prosperity is the ratio of mill workers to the population. Prosperity is further affected by the personal tax rate. The expression (110 - personal tax rate) * .01 means prosperity decreases with a higher tax rate.

If P is less than 1, the population decreases due to emigration, a lower birth rate, and a higher death rate. If P reaches a zero, the entire population dies, and the player loses. If, on the other hand, P is greater than 1, population increases due to immigration, a higher birth rate, and a lower death rate. If P = 1, status quo is maintained.

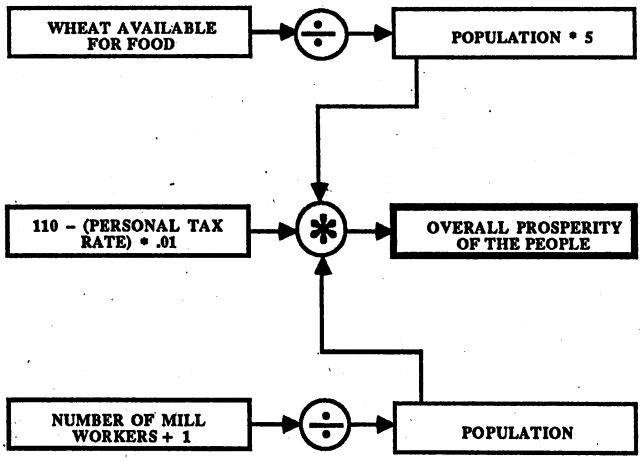
If the food supply exceeds 5 bushels of wheat per person, the prosperity value increases. If the food supply drops below this limit, prosperity decreases. The ratio of mill workers to the population affects prosperity in a similar manner: If less than 10% of the population is working in the mills, prosperity drops below 1. If the personal tax rate is above 10%, prosperity decreases. If the rate is below 10% prosperity increases.

LISTING ANNOTATIONS

Line Nos.	
100-190	Program header
200-260	Title screen
270-310	Initialize program
320-370	Get players' names
380-420	Main control loop
430-510	Main data screen
520-620	Display map screen, get key press
630-740	Cursor movement routines
750-1040	Add unit of land
1050-1130	Add mill or commerce
1140-1220	Wheat option main menu
1230-1260	Feed people
1270-1300	Plant seed
1310-1340	Put wheat in storage
1350-1380	Buy wheat
1390-1420	Sell wheat
1430-1500	Input routine
1510-1520	Bad input messages
1530-1680	Hire/fire soldiers routine
1690-1820	Set tax rate routine
1830-2360	Year-end harvest report
2370-2490	End-of-game routine
2500-2650	Display main data screen
2660-2930	Initialize program
2940-3040	Load old game
3050-3140	Save game
3150-3170	Check disk error channel
3180-3220	Play-again option
3230-3260	Routine to shift and display text

DESIGN FOCUS

Serf City Prosperity



DIRECTORY OF VARIABLES

Variables	Functions
K(,)	Kingdoms' vital statistics array
P$( )	Players' names
A$, B$	Get inputs
DN$	Value of DN for file routine
F$	File name
I1$, I2$	Legal characters
K$	Numeric file access
S$	Text to be displayed
VW$	Value of VW for file routine
W$	Key pressed
YR$	Value of YR for file routine
A	Value of input utility
A0, A1, A2, A3	Constants for 0, 1, 2, and 3
A4, A5, A6, A7	Constants for 4, 5, 6, and 7
A8, A9	Constants for 8 and 9
C	Character under cursor
C1, C2, C3, C4	Characters around cursor
CH	Character placed into map
DN	End-of-game flag
E	Employment in fields
F	Employment in mills
G	Employment in commerce
L	Length of string
P	Prosperity of people
P1	Current player
P2	Opposing player
R	Random number
S	SID chip start address
V	Start of video memory
VW	Value of wheat
W	ASCII value of key pressed
X, Y	Screen coordinates
YR	Current year
Z	Utility loop counter

Serf City

REMARKS

The Design Focus for the Apple version of *Serf City* shows how the program continually updates and displays each kingdom's population. Either kingdom's population can increase or decrease due to births, deaths, immigration, and emigration. The amount of wheat available per person, personal tax rate, and the ratio of mill workforce to total population are some of the other factors that alter population. The variable P (prosperity) contains all of these factors.

We use this formula to calculate the number of births: BIRTHS = INT(K(Z,8) * .05 *P * RND(1)). Here, K(Z,8) represents the current population. Births cannot contribute more than 50% of a year's population growth. The next step in calculating population is to determine the number of deaths: DEATHS = INT(K(Z,8) * .2 / P * RND(1)). (The numbers .05 and .2 in these equations are scaling factors.) If deaths exceed current population, the end result will be zero, and the kingdom will die.

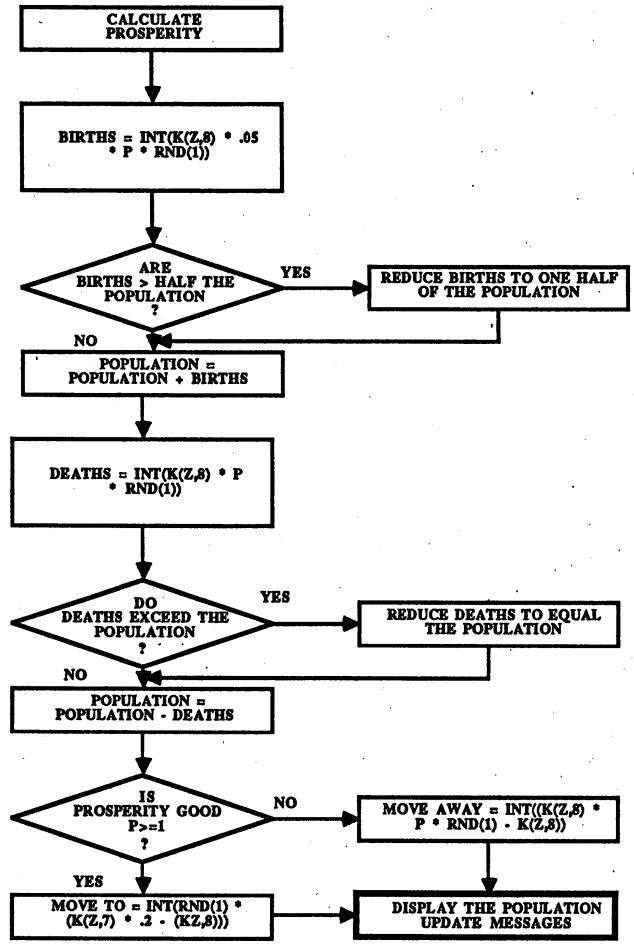
Finally if P is greater than or equal to 1 ("even" or good prosperity), the program calculates the number of people that moved into the kingdom during the year. But if P is less than 1(poor prosperity), the program calculates how many people moved away from the kingdom: 1) MOVE TO = INT(RND(1) * K(Z,7) * .2 - (KZ,8)). 2) MOVE AWAY = INT((K(Z,8) * P * RND(1)) - K(Z,8)) Here, K(Z,7) represents the amount of food available. Adding immigrants to, or subtracting emigrants from, the previous calculations yields the updated figure.

LISTING ANNOTATIONS

Line Nos.	
100-200	Program header
210-280	Title screen
290-310	Reload above high-resolution screen
320-380	Initialize program
390-400	Get option to include sound effects
410-430	Get option to load previous game
440-490	Get players' names
500-620	Main data screen and options
630-800	Map screen, get options
810-920	Add unit of land to map
930-1030	Resolve conflict
1040-1070	Add mill to map
1080-1110	Add commerce to map
1120-1150	Put character on screen
1160-1240	Wheat menu
1250-1270	Feed people
1280-1300	Plant seed
1310-1330	Store wheat
1340-1360	Buy wheat
1370-1390	Sell wheat
1400-1420	Input routine
1430-1450	Bad input messages
1460-1610	Hire/fire soldiers
1620-1730	Set tax routine
1740-2150	Year-end harvest routine
2160-2270	Display the main data screen
2280	Sound routine
2290-2300	Key scan routine
2310-2350	Option to save game
2360-2380	Option to continue game
2390-2490	Exit game, show results
2500-2590	Load old game
2600-2670	Save game to disk
2680-2780	Error routine
2790	Data for initial kingdom
2800-2820	Initial data for map screen
2830-2890	Misc. program data

DESIGN FOCUS

Serf City Population Growth



DIRECTORY OF VARIABLES

Variables	Functions
P$( )	Players' names
S$( )	Map screen
K(,)	Kingdom information
A$, B$	Input routine
A	Value returned from input routine
C	Character under cursor
C1, C2, C3, C4	Characters around cursor
DN	End-of-flag routine
E, F, G	Employment levels
OFF	Turns off graphics routine
P	Prosperity of people
P1	Current player
P2	Opposing player
PD	Prodos flag
PRNT	Graphics print routine address
R	Random number
SOUND	Sound-effects routine address
VW	Value of wheat during year
X, Y	Screen coordinates
YR	Current year of simulation
Z	Utility loop counter

Serf City

REMARKS

Serf City wouldn't be realistic without enabling the kingdoms' armies to battle for land ownership. The IBM routine outlined in the Design Focus depicts the process of combat resolution—how the program decides which army wins.

To determine if combat is possible, the program first uses the SCREEN function to get the colors of the characters next to the cursor and stores them in the variables C1, C2, C3, and C4. In this way, it determines whether the cursor is next to the player's own land. If so, a routine checks the color of the character beneath the cursor. If it is the opposing player's color, the program branches to a routine to resolve the conflict.

The number of soldiers, the amount of land, and the pay per soldier form a measure of each player's strength going into the battle. The program uses the following equation to determine the battle outcome:

```
P = K(P1,9) * K(P1,10) / K(P1,1) - K(P2,9) * K(P2,10) / K(P2,1)
```

ATTACKER	DEFENDER	DATA TYPE
K(P1,1)	K(P2,1)	Total land owned
K(P1,9)	K(P2,9)	Number of soldiers
K(P1,10)	K(P2,10)	Soldier's pay

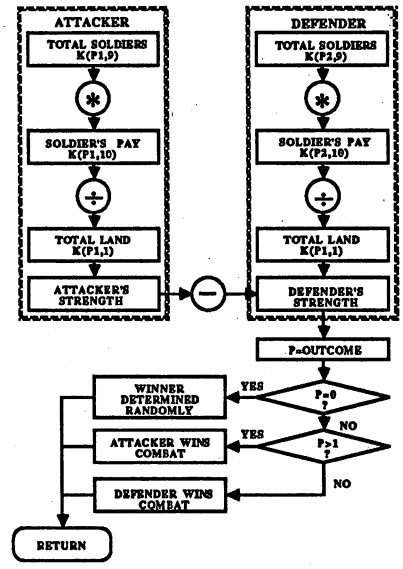
For each player, the kingdom's total number of soldiers is multiplied by the pay per soldier and then divided by the kingdom's total land. The resulting "strength" of the defender is subtracted from the strength of the attacker to set the value of P. If P is equal to a number greater than zero, the attacker wins. If P is less than zero, the defender wins. If the value of P is *zero*, a tie is indicated. In this case, the program determines the winner randomly.

LISTING ANNOTATIONS

Line Nos.	
100-240	Program header
250-310	Initialize program
320-360	Load saved game
370-440	Get players' names
450	Set up playing screens
460	Main control loop
470-540	Main data screen options
550-690	Start Map option
700-850	Add land
860-900	Add mill
910-950	Add commerce
960-1020	Resolve conflict
1030-1130	Display main data screen
1140-1260	Initialize map screen
1270-1430	End-of-game routine
1440-1550	Build wheat option screen
1560-1760	Wheat options routine
1770-1810	Bad input message
1820-1920	Build soldiers screen
1930-2070	Soldiers option routine
2080-2200	Build tax screen
2210-2310	Tax option routine
2320-2810	Year-end harvest routine
2820-2870	Display bold box
2880-2920	Display box
2930-3120	Input routine
3130	Initialization data
3140-3160	Main data screen data
3170	Get file name
3180-3230	Error trapping routine

DESIGN FOCUS

Serf City Combat Resolution



DIRECTORY OF VARIABLES

Variables	Functions
K(,)	Player information array
P$( )	Players' names
S$( )	Main data screen array
A$	Format string for PRINT USING
B$	40 spaces
E$	Error message text
FIL$	File name
I$	Test for key press in error routine
K$	Return key pressed in input routine
MS$	Invalid input message
S$	String returned from input routine
TST$	Illegal-file-name characters
A1	Returns character from screen
A2	Returns color of screen location
C	Player color
C1, C2, C3, C4	Characters around cursor
CS	Input string length
D	Utility
DD	Delay counter
DN	Game over flag
E	Employment in fields
EL	Line where error occurred
ER	Error code
F	Employment in mills
G	Employment in commerce
H, W	Height and width of box
H1, H2, V1, V2	Box-edge characters
I	Utility
L	Length of string to get
LL	Lower-left corner of box
LR	Lower-right corner of box
P	Utility
P1	Current player
P2	Opposing player
R	Random number
SP	Position in input routine
ST	Error return flag
T	Temporary variable
UL	Upper-left corner of box
UR	Upper-right corner of box
VW	Value of wheat
X, Y	Screen coordinates
YR	Current year of game
Z	Loop counter for year-end harvest

Serf City

REMARKS

In the Atari version of this simulation, there are two types of tax—a personal tax placed on a kingdom's population, and a commerce tax placed on a kingdom's mills and commerce. The program uses a "prosperity variable" as a measure of the kingdom's wheat available per person, the personal tax rate, and the ratio of mill workforce to total population. The adjacent Design Focus depicts how prosperity and other variables combine to determine both the kingdom's personal tax income and its commerce tax income.

The personal tax affects all employed citizens in the kingdom. To calculate the personal tax, we first add the number of people working in fields, mills, and commerce.

```
TOTAL EMPLOYED = FIELDS + MILLS + COMMERCE
```

We then multiply this value by the prosperity variable, by the personal tax rate, and by a scaling factor.

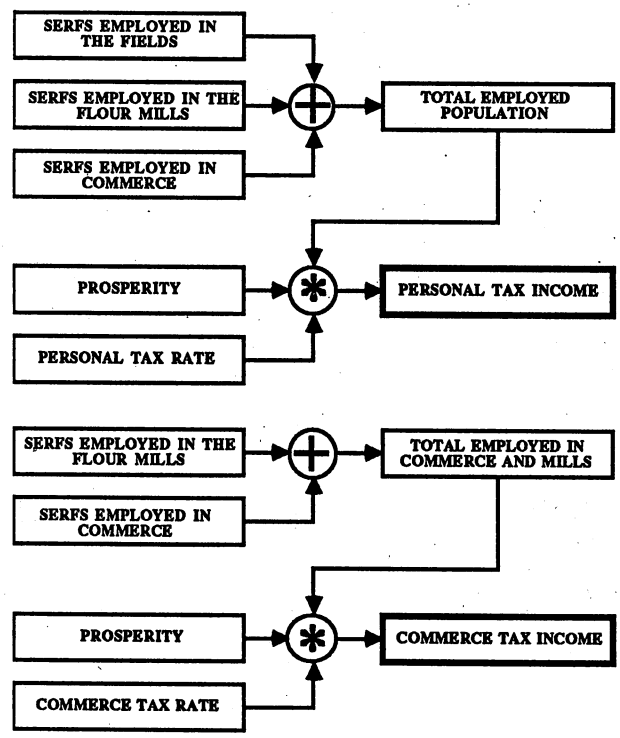
```
PERSONAL TAX INCOME = EMPLOYED * TAX RATE * PROSPERITY * .15
```

The number of people employed in mills and commerce determines the commerce tax income. These two workforces combine into a total, which the program uses as one variable to determine the kingdom's commerce tax income. As with the personal tax, the prosperity variable also figures in the commerce tax equation. The following expression shows how we multiply the number of workers in the mills and commerce by the prosperity variable, commerce tax rate, and a scaling factor to arrive at the commerce tax income:

```
COMMERCE TAX INCOME = EMPLOYED * TAX RATE * PROSPERITY * .25.
```

DESIGN FOCUS

Serf City Tax Structure



LISTING ANNOTATIONS

Line Nos.	
100-190	Program header
200-270	Initialize program
280-290	Title screen
300-320	Get option to load old game
330-380	Get players' names
390-410	Main control loop
420-490	Main data screen and options
500-660	Map screen and options
670-810	Add unit of land to map
820-860	Add commerce to map
870-910	Add mill to map
920-1020	Resolve conflict
1030-1110	Wheat option screen
1120-1140	Feed people
1150-1170	Plant seed
1180-1200	Store wheat
1210-1230	Buy wheat
1240-1260	Sell wheat
1270-1320	Input routine
1330-1340	Bad input messages
1350-1520	Hire/fire soldiers
1530-1580	Set tax rate routine
1590-1980	Year-end harvest report
1990-2110	End-of-game routine
2120-2270	Load and save routine
2280-2380	Display main data screen
2390-2460	Program data
2470-2600	Initialization routine
2610-2680	CIO routine for disk and tape operations

DIRECTORY OF VARIABLES

Variables	Functions
K(,)	Kingdom's vital statistics array
P1$, P2$	Players' names
K$	Legal keys
A$, B$	Input routine utility variable
ML$	Machine language routine
S$	Map screen
A	Value returned from input routine
ADD, ADH, ADL	For CIO routine
CHN, DL, IOCB	For CIO routine
NMH, NML NUM	For CIO routine
C	Character under cursor
C1	Utility for calculations
C2	Utility for calculations
C3	Utility for calculations
CH	Character placed into map
DN	End-of-game flag
E	Employment in fields
F	Employment in mills
G	Employment in commerce
P	Prosperity of people
P1	Current player
P2	Opposing player
R	Random number
RWF	For CIO routine
VW	Value of wheat during year
W	Key returned from key scan
YR	Current year of simulation
Z	Utility loop counter

Cell Mates

REMARKS

One of the most complex algorithms in the TI version of *Cell Mates* is the one that determines the level of protein. Protein can either be produced by the ribosomes, or it can be imported from outside the cell. At the start of the simulation, the protein level is at 50% of maximum. The protein supply must reach this maximum for the cell to reproduce. It is not easy to double the supply of protein, however, because protein must be exported. The Design Focus illustrates the algorithm that we use to control these processes.

Ribosomes are the primary source of protein. Each ribosome can produce a limited amount of protein, however, so the number of ribosomes present determines the amount of protein that can be produced in any given period. This control factor is represented in the Ribosome Level parameter in the flow chart. The routine multiplies the number of ribosomes by the Protein Energy (the amount of ATP available for protein production) and calculates the square root of the result to arrive at the optimum protein production value. The routine then takes other controlling factors into consideration. These factors include the cell's age, the internal temperature deviation from optimum, and the amount of available nutrients—all of which have a detrimental effect on the cell's ability to produce protein. The routine multiplies the optimum increase in protein production by adjustments based on these three factors to determine the increase in protein level.

The routine then adds the present protein import and executes a check on the result for the maximum and minimum protein increases in order to avoid unrealistic results.

The value we end up with after all of this calculating is the amount of change that is to take place in the target protein level, or the level that protein would reach if change in level were instantaneous. The routine adds this value to the present target protein level to determine the new target protein level.

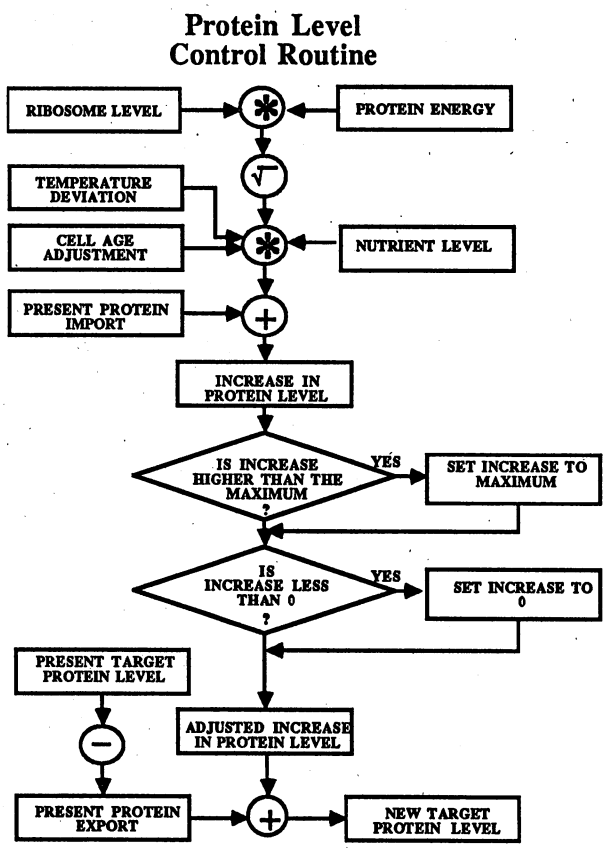
Three equations contain the entire algorithm and are located in lines 440, 470, and 480.

HCM Glossary terms: ATP, ribosomes

LISTING ANNOTATIONS

Line Nos.	
100-190	Program header
200-240	Program initialization
250-280	Main control loop
290	Branch to end-of-program routine
300-320	Option to exit game
330-480	Vital statistics calculations
490-530	Check vital status
540-670	Cell dies from various causes
680-710	Cell reproduces
720-730	Option to play again
740-770	Update meter positions
780-790	Update mitochondrion motion
800-850	Administer ATP
860	Key scan routine
870-950	Character shape data
960-1020	Screen format data
1030-1040	Initial values for the C( ) array
1050-1090	Playing screen set-up
1100-1110	Initialize variables

DESIGN FOCUS



DIRECTORY OF VARIABLES

Variables	Functions
C( )	Graphics information
A$	Input routines
I$	Legal keys
A	Age deviation factor
AGE	Age of the cell
ATP	Amount of ATP administered
D	Flag for reason for game termination
DST	Destination of mitochondrion
DX	X coordinate of mitochondrion's destination
DY	Y coordinate of mitochondrion's destination
K	ASCII value of the key pressed
MF	Motion indicator
MX	The speed of the mitochondrion on the X axis
MY	The speed of the mitochondrion on the Y axis
P	The position of the key pressed within the list of legal keys
RN	Random factor: nutrient supply
RT	Random factor: internal temperature
ST	Keyboard status
TA	Temperature deviation multiplier
Z	Utility loop counter

Cell Mates

REMARKS

One of the most challenging tasks on any computer, including the Commodore 64, is simulating the external environment—the physical world is very complex, to say the least.

To make our programs more realistic, we often build in algorithms which simulate real events. One such event is the transfer of energy from one place or process to another. This transfer would be fairly simple to program if energy transfer were instantaneous. In almost all cases, however, this transfer takes place over a period of time.

In *Cell Mates*, energy transfer occurs in all processes; heat generation, protein production, etc. To handle this intricate pattern of energy transfer, we created the algorithm illustrated by the chart on this page. This algorithm simulates the real-time delay that affects almost any process-control device, whether it be industrial or biological.

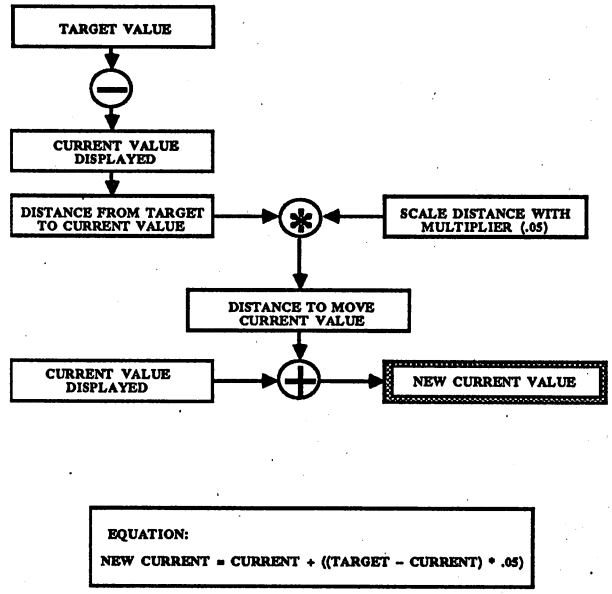
The algorithm first determines the value that would occur if the energy transfer were instantaneous. This is the target value. The current value (the value displayed on the indicator) slowly approaches the target value. If the target value is not affected by other processes that influence energy transfer, such as the age of the cell, the current value eventually reaches the target value.

We employ a multiplier (.05) to control the speed at which the current value approaches the target. By applying this multiplier to the distance between the current value and the target value, we get a fraction of the distance between the current value and the target value. ((target value - current value) * .05). That fraction constitutes the distance that the current value increments as it approaches its target.

The beauty of this routine is that the distance between the two values determines how quickly the current value approaches the target. The farther the current value is from the target value, the faster the current value approaches the target value.

DESIGN FOCUS

Process Control
Delay Simulation



LISTING ANNOTATIONS

Line Nos.	
100-200	Program header
210-290	Initialize program
300-320	Main control loop
330-350	End of game—option to play again
360-600	Vital statistics calculations
610-630	Update indicators
640-680	Move mitochondrion
690-740	Update ATP levels
750-780	Update age counter
790-820	Read and decode keypress
830-890	Sound effects routines
900-960	Check cell's health
970-1170	Print end-of-game message
1180-1540	Initialize program graphics and screen
1550-1560	Data for C( ) array initial values
1570	Keypress translation data
1580	Indicator highlights data
1590-1600	Custom indicator character data
1610-1780	Sprite shape data
1790	Screen column data
1800-1810	Exception characters
1820-1840	Indicator labels
1850-1930	Cell model data
1940-1970	Exit program, option to play again

DIRECTORY OF VARIABLES

Variables	Functions
C( )	Values for indicators
KEYS( )	Key to location translation
METR( )	Screen offset to start of indicator
MHST( )	Offset to pointer position
A$	Message displays
K$	Inputs
A	Age deviation factor
AGE	Age of the cell
ATP	Amount of ATP administered
BASE	Character for pointer
BCKGR	Screen background color (0)
BRDR	Screen border color (4)
CBASE	Custom character set location
CCOL	Screen color (12)
D	Flag for reason for game termination
I, II	Utility—loop counters
K	Key pressed
MCHAR	Starting character for pointers
MIT	Motion indicator
NDX	Screen offset
RN	Nutrient level random multiplier
RP	Position of pointer in indicator
RS	Pointer character
RT	Random multiplier for temperature
SBASE	Start of sprite shape memory
SND	Sound routine address
TA	Temperature deviation multiplier
VID	Start of screen memory
X	Utility variable
Z	Utility variable

Cell Mates

REMARKS

The main control loop is the heart of the Apple version of *Cell Mates*. Simulations and games usually have far more intricate control routines than other programs, because of the complexity of the interaction taking place. Often these programs operate continuously, requiring the control loop to update and maintain certain conditions in an apparently instant process.

Because a simple routine is much easier to write and debug than a complex routine, normally the first rule in designing a control routine is to keep it simple. Thus, all specialized functions of the program should be kept in separate routines away from the main control routine.

For the sake of speed, however, there are times when it is better to leave some code in the main control routine, rather than in a separate routine. In *Cell Mates*, the key-scan and control code is the hub from which all other actions and reactions emanate. It therefore makes more sense to place this code in the main control routine.

The main control routine starts by checking whether a key has been pressed. If no key has been pressed, the program skips the rest of the key-scan code and continues with the control loop.

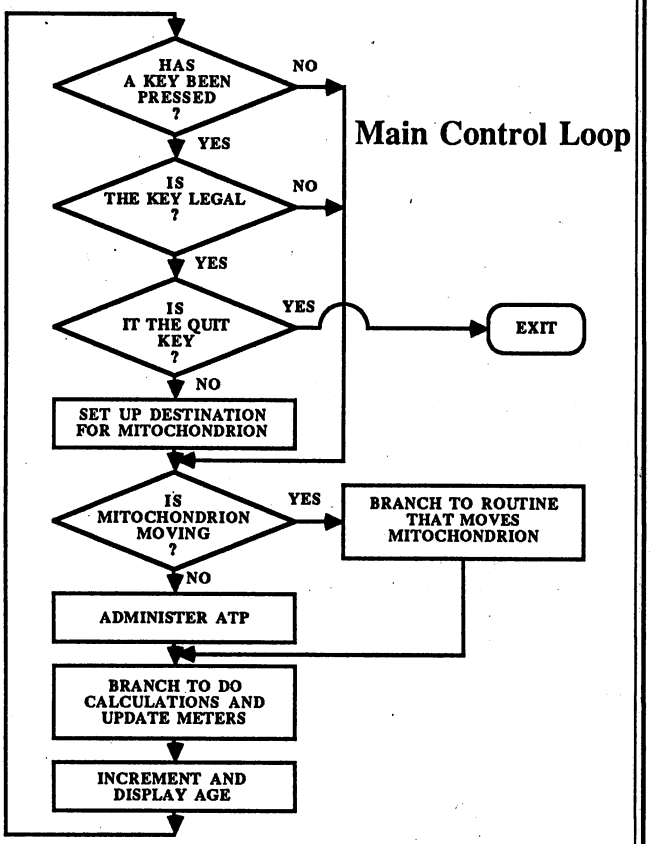
If a key *has* been pressed, the program checks to see whether the key is legal. If the key is legal, the program continues, otherwise the program branches around the rest of the key-scan code as if no key had been pressed. The routine now checks to see if the selected key is "Q" or "q". If so, the program branches to the exit routine. If a different legal key has been pressed, the routine uses that key as an indicator of the mitochondrion's destination.

The control routine then checks the MF flag to see if the mitochondrion is supposed to be moving. If it is, the program calls the routine that moves the mitochondrion; otherwise ATP is administered. The routine then performs a branch to do all of the cell's calculations. Finally, the main control routine increments the cell's age and displays the update on the screen.

LISTING ANNOTATIONS

Line Nos.	
100-200	Program header
210-260	Misc. subroutines
270-300	Load program above high-res screen
310-330	Program initialization
340-360	Sound effects option
370-450	Main control loop
460-490	End of game, option to play again
500-690	Vital statistics calculations
700-760	End-of-game check
770-910	End-of-game subroutine
920-950	Pointer movement
960-980	Mitochondrion movement
990-1050	ATP administration
1060-1140	Sound effects routines
1150-1400	Game screen graphics
1410-1550	Initialize variables
1560-1590	Data for the graph labels
1600-1650	C( ) array data
1660-1720	CALL PRNT routine hexadecimal data
1730-1760	Data for the high-resolution numbers
1770-1840	Data for the high-resolution characters
1850-1880	Sound and shape table data
1890	Data statement error message

DESIGN FOCUS



DIRECTORY OF VARIABLES

Variables	Functions
A$	Inputs and reading data
I$	List of legal key presses
C( )	Values for indicators
OLD( )	Old indicator values
P( )	Graphics coordinates
A	Age deviation factor
AGE	Age of the cell
AP	The amount of ATP administered
D	Flag for reason for game termination
DST	Destination of mitochondrion
DX	X coordinate of mitochondrion destination
DY	Y coordinate of mitochondrion destination
HIGH	Upper limit used by LIMIT function
K	Key pressed
LOW	Lower limit used by LIMIT function
NOISE	NOISE effects routine address
MF	Motion indicator
MX	Movement along X axis
MY	Movement along Y axis
OFF	Address of sound routine
P	Key index variable
PRNT	Address of PRNT graphics routine
RN	Random nutrient multiplier
RT	Random temperature multiplier
SOUND	Address of SOUND effects routine
TA	Temperature deviation multiplier
X1	Utility variables
X2	Utility variables
Y	Utility variables
Z	Utility variables

Cell Mates

REMARKS

Paint tiling on the IBM version of *Cell Mates* is a pleasant bonus to the graphics commands already built into IBM and Tandy BASIC. Paint tiling is an option of the PAINT command that allows specification of the pixel pattern and colors to be used in the filling process. This useful option allows the programmer to design shape interiors and to "dither" several colors together for added graphics effects.

Paint tiling requires a string expression that specifies the pixel and color patterns. The width of the tile area is one byte (eight bits), as shown in Chart 1 of the Design Focus. To the right of this diagram are the hexadecimal values of each byte. Bits 0 through 3 make up one hexadecimal character; bits 4 through 7 constitute the other hexadecimal character.

The resolution of the screen determines how many pixel columns that the tile can contain. In medium-resolution graphics mode 1 (SCREEN 1), it takes two bits to define a color. Dividing 2 into the bit width of 8 gives us a pixel width of 4—so we can paint a tile that is 4 pixels wide. Chart 2 illustrates a medium-resolution tile; Chart 3 shows the effect when 4 of these tiles are placed on the screen.

In medium-resolution mode there are three colors available for the foreground, and one color for the background. The chart below lists the colors that are available from palette 1 of the COLOR command (see your BASIC reference manual for more information on the COLOR command).

Bit Pattern	Hexadecimal Character	Colors Produced	
0 0 0 0	0	Background	Background
0 0 0 1	1	Background	Green
0 0 1 0	2	Background	Red
0 0 1 1	3	Background	Brown
0 1 0 0	4	Green	Background
0 1 0 1	5	Green	Green
0 1 1 0	6	Green	Red
0 1 1 1	7	Green	Brown
1 0 0 0	8	Red	Background
1 0 0 1	9	Red	Green
1 0 1 0	A	Red	Red
1 0 1 1	B	Red	Brown
1 1 0 0	C	Brown	Background
1 1 0 1	D	Brown	Green
1 1 1 0	E	Brown	Red
1 1 1 1	F	Brown	Brown

Background color is specified with the COLOR statement

HCM Glossary terms: dither, tile.

LISTING ANNOTATIONS

Line Nos.	
100-240	Program header
250-280	Program initialization
290-380	Main control loop
390-440	Cell limits check
450-650	End-of-game option to play again
660-820	Mitochondrion movement
830-1030	Cell conditions update
1040-1190	Screen indicators update
1200-1420	Variable initialization
1430-1530	Screen initialization
1540-1590	Cell graphics
1600-1680	Internal parts graphics
1690	Indicator graphics
1700-1710	Data for green area of indicators
1720-1750	Indicator descriptions
1760-2080	Program data

DESIGN FOCUS

Fancy Graphics With Paint Tiling

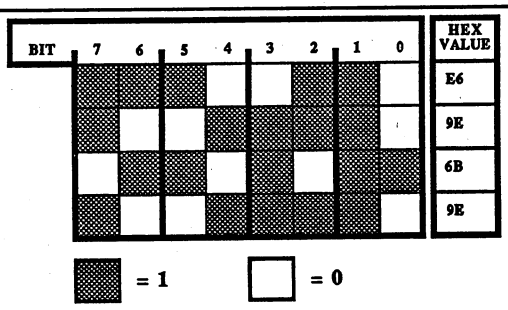


CHART 1—Bit Patterns And Hexadecimal Code

COLOR PATTERN

3	2	1	2
2	1	3	2
1	2	2	3
2	1	3	2

1=GREEN
2=RED
3=BROWN

CHART 2—Color Tile Pattern

EFFECT ON SCREEN

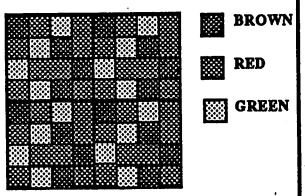


CHART 3—Color Screen Pattern

DIRECTORY OF VARIABLES

Variables	Functions
C(,)	Internal cell coordinates
MITO%( )	Mitochondrion shape array
P( )	Indicator shape array
A$	Screen display
AA$	Indicator graphics
B$	Indicator graphics
D$	Indicator graphics
I$	List of legal keys
K$	Utility
A	Age deviation factor
AGE	Age of cell
ATP	Amount of ATP administered
B	Utility
D	Utility
D1	Utility
D2	Utility
CC	Distance for mitochondrion to move
DONE	End of game
DST	Destination of mitochondrion
DX, DY	Mitochondrion motion
EX.E	Export energy
EX.L	Export protein level
EX.TEMP	External temperature
GX, GY	Coordinates of destination
IN.TEMP	Internal temperature
IPRO	Input protein
MF	Mitochondrion status
MITO.X	X coordinate of mitochondrion
MITO.Y	Y coordinate of mitochondrion
NM	Mitochondrion motion flag
NUT.L	Nutrient level
PRO.E	Protein energy
PRO.L	Protein level
RIBO.E	Ribosomes energy
RIBO.L	Ribosomes level
RN	Random nutrient adjustment
RT	Random temperature adjustment
TA	Temperature adjustment
WASTE.E	Waste energy
WASTE.L	Amount of waste in the cell
X, X1, X2	Utility variables
Y, Y1, Y2	Utility variables
Z	Utility variables

Cell Mates

REMARKS

The Atari version of *Cell Mates* uses the Atari computer's ability to place machine-language routines in a string variable. Although many machine-language routines are written for a particular section of memory (because they access particular addresses within the routine), routines that use the string variable method *must* be relocatable.

The routine that moves the mitochondrion vertically is located in the string ML2$ in line 330. This line initializes the length of the string to 30 characters, then places 30 characters in the string. The ASCII values of these characters actually make up the code for the machine-language routine. The Design Focus demonstrates how these ASCII characters correspond to the machine-language instructions.

To branch to such a routine, we need to know the memory address of the string. Atari BASIC offers a convenient way to retrieve the address of the string—the ADR function. In line 1120, ADR(ML2$) returns the address of the string ML2$. In this way, the USR function transfers program control to the machine-language routine:

```
H=USR(ADR(ML2$),256+SGN(MY-DY))
```

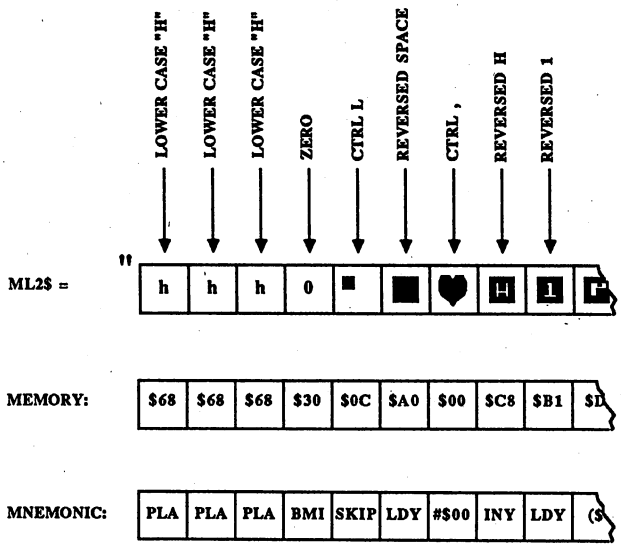
The variable H is a dummy variable required for BASIC syntax—any variable name could be used in place of H. The first parameter in the USR function is the address of the string ML2$ returned from the ADR function. The second parameter is a numeric expression passed to the machine-language routine for it to determine the direction that the mitochondrion moves. The low byte of this value is either a 1 or a 255. The machine-language routine interprets these values as positive or negative, respectively. A negative value moves the mitochondrion down; a non-negative value moves it up.

The USR function passes this value to the machine language routine by placing it on the stack along with the number of parameters being passed and the return address of the BASIC program.

HCM Glossary terms: parameter, stack.

DESIGN FOCUS

Atari
Machine Language Strings



LISTING ANNOTATIONS

Line Nos.	
100-200	Program header.
210-250	Title screen
260-400	Program initialization
410-470	Main control loop
480-530	End of program—branch to appropriate message routine
540-770	Vital statistics calculations
780-840	Check for an end-of-program condition
850-1050	End-of-game messages, option to play again
1060-1090	Meters update
1100-1130	Mitochondrion movement
1140-1210	ATP administration
1220-1370	Character graphics data
1380-1430	Cell data
1440-1460	Data for initial values of C( ) array
1470-1760	Playing screen graphics
1770-1780	C( ) array initialization
1790-1840	Limit values to minimum and maximum
1850-1920	Death-of-cell special effects routines

DIRECTORY OF VARIABLES

Variables	Functions
C( )	Values for indicators
A$	Utility string for inputs etc.
ML$	Routine to move the character set
ML2$	Routine to move the mitochondrion
A	Utility variable
AGE	Age of the cell
ATP	Amount of ATP administered
CH	Rebuilds the character set
D	Flag indicating when game is over
DST	Destination of mitochondrion
DX	X coordinate of mitochondrion destination
DY	Y coordinate of mitochondrion destination
H	Starts the ML2$ Routine
I	Utility loop counter
J, K	Indicates element in C( ) array for limit checking
MF	Mitochondrion movement flag
MX	Mitochondrion X axis movement
MY	Mitochondrion Y axis movement
P	Key pressed
PM	Utility variable
RN	Random multiplier affecting nutrient supply
RT	Random multiplier affecting the internal temperature
TA	Effect that the internal temperature deviation has on other cell parts
X	Used in waste calculation
Y	Used to read data
Z	Utility — loop counter

NanoAssembler

REMARKS

The TI version of the *NanoEditor* uses the POS function to control program flow when the user selects a particular editing command. The POS function searches through a string for a smaller string within it, and returns (gives) the numeric value of the POSition of the smaller string. In this way, line 430 sets OP equal to the POSition of the S$ (the user's selected command) in CMD$ (the legal commands—"ADIEL").

As shown in the Design Focus, if OP is not a 1, the program prompts the user for the number of the line to be Deleted, Edited, Inserted, or Listed (see lines 460-600 in the program). If OP is set to 1 (for the Add command), the program already knows the line number to be added, so it branches around this section. In either case, OP then controls the ON-GOSUB in line 610 according to the chosen command.

HCM Glossary terms: branch, string.

LISTING ANNOTATIONS — *NanoEditor*

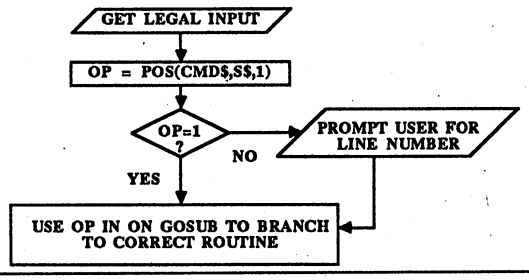
Line Nos.	
100-190	Program header
200-230	Initialize program
240-290	Main menu
300-620	Main editing loop
630-830	Add a line
840-950	Delete a line
960-1140	Edit a line
1150-1340	Insert a line
1350-1570	List lines
1580-2010	File routines
2020-2140	Print-lines routine
2150-2240	End program
2250-3100	Input routine
3110-3190	Print-menu routine
3200-3220	Assure number is three digits
3230-3270	Set parameters for input routine
3280-3310	Print line number
3320-3360	Menu data
3370-3400	Read keyboard
3410-3430	Delay loop
3440-3470	Initialize-program subroutine

LISTING ANNOTATIONS — *NanoAssembler*

Line Nos.	
100-190	Program header
200-270	Initialize program
280-450	Program driver
460-820	First pass
830-1120	Second pass
1130-1170	Get keypress and return to main driver
1180-1310	Error routines
1320-1460	File routines
1470-1520	Prepare assembler output
1530-1590	Strip line of remarks and spaces
1600-1740	More file routines
1750-1820	Look for a legal mnemonic
1830-1930	Set up printer
1940-1960	End program
1970-2700	Check mnemonics and directives
2280-2330	Convert a number from binary
2340-2390	Convert a number from hexadecimal
2400-2700	Evaluate data field
2710-2830	Menu routine
2840-2880	Get a Y or N keypress
2890-2950	Delay loop
2960-3030	Initialize-program subroutine
3040-3050	Mnemonics and assembler directives

DESIGN FOCUS

Using the POS Function For Program Flow



DIRECTORY OF VARIABLES — COMMON

Variables	Functions
A	Store the tab position
A$	Utility variable
FL$	File name
I,IT	Loop counters
K	Keyboard-input variable
LI$( )	Array that stores source-code lines
LN	Current line of text
NI	Number of items in current menu
NL	Current number of source-code lines
PR$	Printer parameter
S	Keyboard status

DIRECTORY OF VARIABLES — *NanoEditor*

Variables	Functions
B	Lowest legal ASCII value for input
C	Screen column position during input
C$,CN$	String of possible ASCII inputs
CMD$	Possible editing commands;"ADIEL"
DC	Disk/Cassette flag
E	Position of last character input
ESC	Flag used for escape key—(FCTN) 9
G	Character read from screen
L	Maximum length of input
NS	Used in input routine
OP	Option chosen from editing prompt
R	Screen row position during input
RW	Limit on row postion
S$	Utility string
SL	Save/Load flag
T	Highest legal ASCII value for input
X	Current cursor postion
CA,KS,LN$,Y,Z	Utility variable

DIRECTORY OF VARIABLES — *NanoAssembler*

Variables	Functions
AD( )	Address array
CA	Current address
EA	Ending address of a line of code
HX$	String of hex characters
LB	Current label number
LB$( )	Array of labels
LFLAG	Flag used to spot label
LL	Length of source-code string
N$	Used to convert numbers to decimal
NM$	Legal numeric-character string
NN( )	Number of nibbles for mnemonics
OP	Current mnemonic (or op-code)
OP$( )	Array of mnemonics (or op-codes)
OT$	Output string
OU$	Output string
PR	Printer flag
T$	Temporary string
TL$	Current source-code line
V	Value of data in right-most field
X,Y	Utility variable

NanoAssembler

REMARKS

A vital task of the *NanoAssembler* is translating numbers expressed in string variables into numeric quantities. The Commodore BASIC VAL function makes this task easy by extracting the numeric value of a decimal number expressed as a string. The Design Focus shows how we use this function to convert binary strings to numeric values in line 1400:

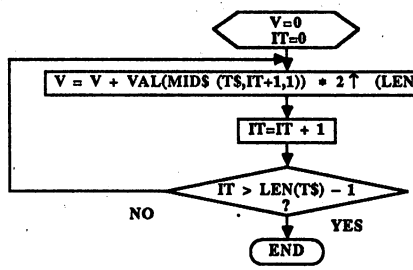
```
1400 V=0: FOR IT=0 TO LEN(T$)-1:
V=V+VAL(MID$(T$,IT+1,1))*2↑(LEN(T$)-IT-1):NEXT
```

The program has already done two things before it branches to this subroutine: It has placed the proper segment (containing the binary number) of the current element of the LI$( ) array into a temporary string T$. Also, it has determined that all characters in T$ are either ones or zeros.

First, V is set to zero. Then, the routine enters a FOR-NEXT loop where the value of IT goes from 0 to the LENgth of T$ minus 1. Thus, each time through the loop, IT+1 points to a different character in T$ so that the routine can use the MID$ function to extract each character of T$. Then the VAL function converts this character into a numeric quantity. Finally, the LENgth of T$ minus (IT-1) raised to the power of 2 results in the correct multiplier for the weight of each bit. The routine adds each value to V in turn, so V contains the correct decimal value when we exit the loop.

HCM Glossary terms: binary, hexadecimal, subroutine, weight.

DESIGN FOCUS



Converting A Binary String To A Numeric Value

DIRECTORY OF VARIABLES – COMMON

Variables	Functions
A$	Utility variable
CA	Current address
CL$,RO$	Clear screen character-CHR$(147)
CO	Current column
CR$	Carriage return-CHR$(13)
DN	File number
DV	Device number
E	Error number
E$	Message from disk error channel
FL$	File name
I,IT,J,RW,X,Z	Loop counters
DC,K,KE,KE$	Key input variables
LI$( )	Source code array
LN	Current line number
NL	Current number of source-code lines
RO	Current row
SA	Secondary address for I/O
TE$	Temporary string for line of source

LISTING ANNOTATIONS – *NanoEditor*

Line Nos.	
100-190	Program header
200-220	Initialize program
230-280	Main menu
290-450	Main editing loop
460-540	Add a line
550-600	Delete a line
610-690	Edit a line
700-770	Insert a line
780-940	List lines
950-1220	File routines
1230-1290	Print routine
1300-1310	End program
1320-1410	Miscellaneous subroutines
1420-1600	Command-line input
1610-1890	Line-input routine
1900-1960	Keyboard-input subroutines

DIRECTORY OF VARIABLES – *NanoEditor*

Variables	Functions
B	Lowest legal ASCII for input
C	Column for cursor
C$, CN$	Legal inputs for command
CM$	Current command
ES	Error number from disk access
FL( )	Array for different tab fields
LL$	Current line length
LN$	Current prompt
LT	Flag for whether a backspace is legal
N	Flag for tab fields
QU	Quote flag
R,RW	Row of input
S$	Temporary variable for source
T	Highest legal ASCII for input

LISTING ANNOTATIONS – *NanoAssembler*

Line Nos.	
100-190	Program header
200-220	Initialize program
230-380	Program driver
390-630	First pass
640-790	Second pass
800-870	Error messages
880-900	Build printer output string
910-940	Truncate source line to remove remark
950-1160	Disk and cassette routines
1170-1200	Get op-code
1210-1240	Set up printer
1250	End program
1260-1670	Subroutines for parsing source code
1680-1710	Keyboard-input routine
1720-1760	Program-initialization subroutines
1770-1780	Keyboard-input subroutines
1790-1800	Sound routines

DIRECTORY OF VARIABLES – *NanoAssembler*

Variables	Functions
AD( )	Array of memory locations
EA	Effective address of next instruction
HX$	String of hex characters
LB	Counter for parsing source code
LB$, T$,TL$	Temporary string for line of source
LF	Load/Save flag (LF=1 means Load)
LL	Line length
N$	Variable for conversion of decimal
NM$	String of legal numeric input
NN( )	Number of nibbles for mnemonics
OP$( )	Array of op-codes and directives
OT$	Output string
OU$	Output string
PR, PR$	Printer device number
V,Y	Utility variables

NanoAssembler

REMARKS

Saving a *NanoAssembler* text file with the same name as one already on disk presents an interesting problem on the Apple computer. As long as the new file you are saving is longer (i.e., takes up more space on disk) than the old one, there is no difficulty. If, however, the file you are saving takes up less disk space, you should DELETE the old file, or the data that extends beyond the end of your new file will remain at the end of the file.

If you DELETE a file without knowing whether the file actually exits, however, you run the risk of a FILE NOT FOUND error, which can halt your BASIC program. An easy way to ensure that the file already exists before you DELETE it, is to OPEN the file—if it didn't exist before you executed the OPEN statement it will afterwards. In DOS 3.3, you may immediately delete the file after you have OPENed it. ProDOS, however, can not DELETE an OPEN file. Therefore, you must CLOSE the file before DELETEing it. The Design Focus details the BASIC syntax that performs the entire procedure.

LISTING ANNOTATIONS — *NanoEditor*

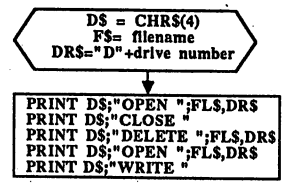
Line Nos.	
100-210	Program header
220-290	Main program loop
300-310	Main menu
320-380	Main editing loop
390-450	Add a line
460-510	Delete a line
520-600	Edit a line
610-670	Insert a line
680-740	List lines
750-780	Enter a line number
790-820	File routines
830-920	Save a file
930-1020	Load a file
1030-1290	Enter file name
1300-1330	Print routine
1340-1360	Turn printer channel on/off
1370-1380	End program
1390-1610	Line-input routine
1620-1660	Set up menu
1670-1770	General menu routine
1780-1810	Get a character from keyboard
1820-1830	Print error messages
1840-1940	Initialize program
1950-2060	Error handler

LISTING ANNOTATIONS — *NanoAssembler*

Line Nos.	
100-210	Program header
220-300	Main program loop
310-500	First pass
510-590	Second pass
600-630	Strip remarks
640-680	Get op-code
690-900	Check mnemonics and directives
910-930	Advance counters and set flags
940-1070	Evaluate data field
1080-1110	Convert a number from binary
1120-1160	Convert a number from hexadecimal
1170-1200	Turn printer channel on/off
1210-1290	Error routines
1300-1410	Load file
1420-1510	Save a file
1520-1780	Enter file name
1790-1820	Get a character from keyboard
1830-1910	Initialize program
1920-2040	Error handler

DESIGN FOCUS

Saving Over An Existing File



DIRECTORY OF VARIABLES — COMMON

Variables	Functions
A$, Z$	Character constants
BL$	CHR$(7), the bell
CR$	CHR$(13), carriage return
D$	CHR$(4), wakes up DOS
DR$	Number of disk drives
ER	Error flag
ESC$	CHR$(27), escape character
ET$( ), GC, IN$	Keyboard-input variables
FL$	File name
HC, HZ, VC	Cursor variables
EI,IT,JI	Loop counter
IX$	Used to get characters from disk file
LF$, RT$	Cursor left, cursor right
LI$( )	Array to hold source file
LN	Current line number
NL	Number of source-code lines
P, XT	Pointers used to evaluate strings
X, X$	Utility variables

DIRECTORY OF VARIABLES — *NanoEditor*

Variables	Functions
B73$	String of 73 spaces
BK$	A single space
CMD$	Possible editing commands; "ADIEL"
DN$,UP$	Cursor down, cursor up
ED$	Legal control characters in edit mode
EOL	Address of EOL routine
ER$	Error message
HZ( )	Horizontal position of cursor
L	Maximum length of input
LO	Length of default string for input
LM	Maximum number of source lines
LV	Departure flag for menu routine
MI	Loop counter for menu routine
MN$( )	Menu options
MN(,)	Screen locations of menu options
RW	Limit on row position
S$	Utility string
TB$	Tab character

DIRECTORY OF VARIABLES — *NanoAssembler*

Variables	Functions
AD( )	Array of memory locations
CA	Current address
EA	Ending address of a line of code
HX$	String of hexadecimal characters
LB	Current label number
LB$( )	Array of labels
LC( )	Label address
LF	Flag used to spot label
LL	Length of source-code line
N$	Used to convert numbers to decimal
NM$	String of legal numeric characters
NN( )	Number of nibbles for mnemonics
OP	Current mnemonic (or op-code)
OP$( )	Array of mnemonics
PR	Printer flag
T$	Temporary string
TL$	Current source-code line
V	Value of data in right-most field
Y,Z	Utility variables

NanoAssembler

REMARKS

IBM PC and PCjr DOS files are easy to access from BASIC. With **WRITE #** and **INPUT #** statements, moving text to and from disk is a relatively error-free process. Only the quote character (") is a major difficulty.

DOS uses this character to delimit the beginning and end of each string as it is saved. Therefore, if you attempt to save a quote character as part of a string, DOS saves only the portion of the string preceding the quote and ignores the rest.

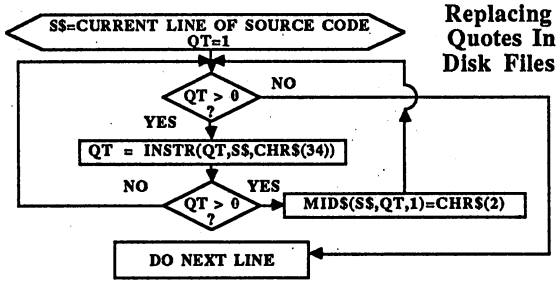
The *NanoEditor* and *NanoAssembler* present a unique solution to this problem. These programs search each string as it is saved and replace the quote with an ASCII 2. (You could use any ASCII character normally not found in text.)

The Design Focus depicts a flow chart of this search-and-replace process. First, a **QT** variable is set to 1 (for the first character in the string). Next, the routine enters a **WHILE** loop where the **INSTR** function searches the temporary string **S$** for quotes. Upon finding a quote, the routine sets **QT** to the quote's location in the string, then replaces it with an ASCII 2. The **INSTR** function then searches the string again, starting at the **QT** location. When there are no more quotes in the string, the **INSTR** function sets **QT** to 0, and the **WHILE** loop ceases.

A very similar routine restores the quotes by searching the strings as the program reads them from the disk.

DESIGN FOCUS

Replacing Quotes In Disk Files



LISTING ANNOTATIONS — *NanoEditor*

Line Nos.	
100-210	Program header
220-270	Initialize program
280-360	Main menu
370-460	Main editing loop
470-510	Add a line
520-570	Parse LI$( ) string
580-650	Get an LI$( ) string
660-680	Get fields from LI$( )
690-740	Display LI$( )
750-860	List lines
870-910	Edit a line
920-960	Delete a line
970-1020	Insert a line
1030-1090	Main file routines
1100-1110	Print-lines routine
1120-1160	End program
1170-1260	Display a box
1270-1550	Input routine
1560-1860	Miscellaneous disk routines

DIRECTORY OF VARIABLES — COMMON

Variables	Functions
A$	Label-field string
B$	Op-code-field string
C$	Comment field string
F$	File name
I,IT,J	Utility counters
INS	Insert-mode status flag
K$	Key input variable
L	Length of input to get
L$	Utility string
LI$( )	Array to hold source file
LN	Current line number
M1	Position of first SP$ delimiter
M2	Position of second SP$ delimiter
M3	Utility variable
MC$	Lowest legal ASCII value for input
NL	Number of source-code lines
OP	Option number for edit routine
P$	Prompt for Get-File-Name routine
S$	String returned by input routine
SP$	Delimiter of LI$( ) strings
SP	Current string position for input
H,H1,H2,LL,LR	ASCII for parts of box
UL,UR,V1,V2	ASCII for parts of box
W	Width of box to be displayed
X	Row number
XC$	Highest legal ASCII value for input
Y	Column number

DIRECTORY OF VARIABLES — *NanoEditor*

Variables	Functions
BB$	String of 40 spaces
CS	Length of input string
F	Field number
LN$	String equivalent of line number
QT	Quote flag
RT	Flag for right-most position in input
SE	Flag for insert routine

LISTING ANNOTATIONS — *NanoAssembler*

Line Nos.	
100-210	Program header
220-250	Initialize program
260	Program driver
270-340	First pass
350-550	Second pass
560-630	Error routines
640-680	Set up rinter
690-700	End program
710-770	Look for a legal mnemonic
780-840	Parse line
850-940	Determine error
950-980	Evaluate op-code
990-1130	Evaluate data field
1140-1400	File routines
1410-1490	Display-box routines
1500-1660	Input routine
1670-1740	Break line into composite strings

DIRECTORY OF VARIABLES — *NanoAssembler*

Variables	Functions
AD%( )	Address array
ASM	Load/Save flag (ASM=1 means Load)
CA	Current addres
D	Utitility variable
DT$	String in Data area left of mnemonic
HX$	Valid hexadecimal characters
LB, LB$, LB$( )	Label variables
LFLAG	Set to one if valid label found
LV( )	Array to contain value of labels
NM$	Legal numeric characters
NN( )	Number of nibbles for an op-code
OP$	String to find mnemonic number
OP$( )	Legal mnemonics
OT$,OU$	Output strings
V	Value of a data string
X1	Row to display source line

NanoAssembler

REMARKS

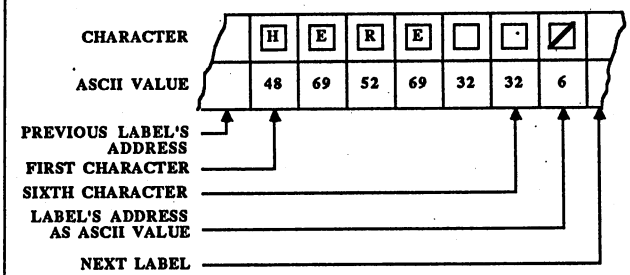
It is essential that the Atari version of the Nano-Assembler assign the correct address to each label it encounters in a source file. We do this by placing all the labels in one string. As the program executes its first pass, it keeps track of the current *NanoProcessor* address in the CA variable. The NN( ) array contains the number of nibbles each op-code requires, and is used to increase CA to keep it current. In this way, whenever the program encounters a label in the Label field, it *knows* the address the label represents.

During the second pass, the program needs to identify each label it encounters in the operand sub-field and match it with the correct address. We devised a way to store all the labels and their addresses in the same string, thus eliminating the need for a separate array of addresses: We place the ASCII that corresponds to the label's address in the string immediately following the label. If the label is less than 6 characters long, the program pads the remaining area with spaces (ASCII 32). Refer to the Design Focus for a graphic representation of this scheme.

The program multiplies the label number kept in the variable LB by 7 to calculate an offset into LB$. Thus, the code to find the first letter of a label is LB$(LB*7,LB*7). Likewise, the address is ASC (LB$(LB*7+6,LB$*7+6)). Because the first label is number 1, and LB$ is DIMed to 175, a maximum of 24 labels is possible, and the first 6 characters of LB$ are not used.

DESIGN FOCUS

The Structure Of The Label String—LB$



DIRECTORY OF VARIABLES — COMMON

Variables	Functions
A, A$,S$	Utility variables
DEV$	I/O device
FL$	File name
I,IT	Loop counters
KEY	Keyboard-input variable
LINE$	String to hold source file
LN	Current line number
NL	Number of source-code lines
X	Error number
Y	Line number that error occurred in

LISTING ANNOTATIONS — *NanoEditor*

Line Nos.	
100-190	Program header
200-230	Initialize program
240-300	Main menu
310-430	Main editing loop
440-530	Add a line
540-590	Delete a line
600-700	Edit a line
710-810	Insert a line
820-940	List lines
950-1270	File routines
1280-1320	End program
1330-1550	Line-input routine
1560-1580	Print menu
1590-1610	Menu data
1620-1770	Miscellaneous subroutines
1780-1960	Error handler

DIRECTORY OF VARIABLES — *NanoEditor*

Variables	Functions
C$	Legal keyboard input characters
CH	Keyboard input character
CMD( )	Indexing into command string
CMD$	Possible editing commands; "ADIEL"
ESC	Flag used for escape key
J	Loop counter
LN$	String containing line number
NM	Number of menu options
RW	Loop counter
TB	Utility variable

LISTING ANNOTATIONS — *NanoAssembler*

Line Nos.	
100-200	Program header
210-240	Initialize program
250-410	Program driver
420-710	First pass
720-930	Second pass
940-1010	Error routines
1020-1220	Miscellaneous subroutines
1230-1240	End program
1250-1490	Check mnemonics and directives
1500-1560	Evaluate data field
1570-1640	Convert from binary
1650-1770	Convert from hexadecimal
1780-1840	Number is decimal
1850-1960	Print menu
1970-2040	Initialize program
2050-2130	Enter file name
2140-2330	Error handler

DIRECTORY OF VARIABLES — *NanoAssembler*

Variables	Functions
AD( )	Array of memory locations
CA	Current address
DELAY	Loop counter
EA	Ending adress of a line of code
EXT$	File name extension
LABLL( )	Array to store length of each label
LB	Index into label string
LB$	Label string
LF	Holds labels
LFLAG	Flag used to spot label
LL	Length of source code line
NI,NN	Utility variables
NN( )	Number of nibbles for mnemonics
OP	Current mnemonic (or opcode)
OP( )	Pointers into opcode string
OP$	String of mnemonics
OT$, OU$	Output string
POWR	Loop counter
PR	Output file number
SQUR	Used to convert from binary
TL$	Current source code line
T$,VU$	Utility string
V	Value of data in right-most field
VLU$	Right-most field

HCM Glossary

VOLUME 5 NUMBER 6

algorithm – A set of rules or procedures used to solve a problem.

amplitude – The height of a waveform measured from peak to peak.

argument – A parameter that is passed by a BASIC function.

ASCII – (American Standard Code for Information Interchange) The computer code most commonly used to represent upper- and lower-case letters, numbers, symbols, and punctuation marks.

assembler – A program that translates or assembles source code into machine language.

ATP – (Adenosine Triphosphate) A product of the mitochondria that serves as the energy source for all cells.

attack – Rate at which a signal reaches its peak level on a traditional ADSR envelope generator.

azimuth – The angular distance of an object measured in degrees clockwise (to the right) from true north.

BCD – (Binary Coded Decimal) A method for representing decimal numbers in binary.

binary numbers – A base-2 numbering system in which the only symbols used are 0 and 1.

bit – (binary digit) The most basic unit of information that the computer uses. Each bit is an electronic impulse, that, combined with other such impulses fed into the computer's circuitry, forms data.

branch – A departure from the sequential execution of program instruction—usually due to the test of a condition.

byte – A sequence of 8 bits used to represent one character.

call – To transfer control to a specified closed subroutine.

component – One of the sine waves that add up to produce a complex periodic waveform.

CONFIG.SYS file – A file on a PC DOS disk that, when DOS boots, configures the way that the operating system views memory, I/O devices, etc.

conjunction – The apparent meeting of two celestial bodies in the same degree of the zodiac.

cross-reference – An organized list of items from several different sources that refers you to the item's folder, card number, or whatever applies.

cursor – A character or mark shown on the video screen to designate where the next character will be printed.

cycle – One complete alternation of a periodic waveform.

decay – Rate at which a signal falls from peak level to sustain level on a traditional ADSR envelope generator.

declination – The angular distance of a celestial body south or north of the celestial equator (the plane of Earth's equator extending to infinity).

dither – To use a combination of dots to simulate a tone in black-and-white or color.

DNA – (deoxyribonucleic acid) Nucleic acids localized in the nucleus of eucaryotic (nucleus-bearing) cells or in the lining of procaryotic (without nuclei) cells as the molecular basis of heredity.

DOS – Disk Operating System.

dummy argument – An argument passed purely to satisfy the syntax of a statement.

ecosystem – An ecological community together with its environment, considered as a unit.

endoplasmic reticulum – A convoluted membrane extending out from the nucleus of a living cell.

envelope – A graphic representation of the envelope generator's output as a function of time.

enzyme – Any of complex proteins produced by living cells that catalyze specific biochemical reactions.

field – A specified area for a particular category of data.

file – A collection of related records (data items), treated as a unit.

flag – A variable or memory location that contains a value that represents a condition the program needs to test.

frequency – The rate of repetition of a waveform. It is perceived as pitch in the audio range.

function – In BASIC, a statement used to return a value. The value often depends on the type of argument passed to it.

fundamental – The principal frequency of a waveform. In the audio range, it is the perceived pitch.

Golgi apparatus – An organelle that packages and transports materials in a living cell to be secreted to the plasma membrane for internal use or expulsion.

hard-code – To specify the values of variables or constants within the program code itself so that they are not subject to change by user input.

harmonic – A component of a periodic waveform. Its frequency is a positive integral multiple of the fundamental frequency.

hexadecimal numbers – A base-16 numbering system using decimal digits 0 through 9 and the letters A through F.

high bit – The left-most bit of a binary number; e.g., the eighth bit of a byte.

horizontal plot position – The horizontal position on the screen at which the next point will be plotted.

index – To selectively choose and organize a group of data (or an organized list of items that refers you to the item's folder, page number, or whatever applies).

interrupt request – A signal that directs the computer away from the program sequence.

iteration – One complete cycle through a process that is repeating.

level of recursion – A routine's depth into a recursive process.

linear – An even progression from one state to another with no deviation in course.

logic operator – A BASIC function used to perform a logical operation.

lysosome – A cell's waste disposal unit—basically a sac pinched off of the Golgi apparatus.

machine language – A native language of the microprocessor in a computer expressed in terms of binary ones and zeros.

mask – A fixed group of characters used to control the retention or elimination of parts of another pattern of characters.

merge – To combine two or more files, or portions of files, into one.

metabolism – The physical and chemical processes of an organism that produce energy and result in the production, maintenance, and destruction of life.

MIDI – (Musical Instrument Digital Interface) Specified protocols for transmitting digital information from one synthesizer, sequencer, computer, signal processor, etc., to another.

mitochondrion – An organelle that produces ATP.

mnemonic – A code or symbol that helps people remember something specific—usually made up of letters from the word or phrases it represents.

nibble – 4 binary digits (bits) of data—a half byte.

noise channel – The channel responsible for producing noises.

nucleus – The central core of a cell containing both the cell's hereditary DNA, and the message-bearing RNA (ribonucleic acid).

organelle – A specialized part of a cell with a specific organ-like function.

overtone – A frequency component of a waveform, that is higher than the fundamental.

op-code – (operation code) The part of a machine instruction that tells the computer which function to perform next.

parameter passing – Sending (giving) a parameter to a particular process.

periodic noise – One of 4 pulse waves produced by the TI-99/4A sound chip.

precession – A gradual change in the angle of the Earth's axis due to "wobbling" in the Earth's rotation.

proper motion – Apparent movement over time of a star in the celestial sphere due to the actual motion of both the star and our sun through space.

protein – Any of several complex compounds that are required for all life processes.

pulse wave – A wave that alternates instantaneously between the two DC volage levels.

pulse width – The percentage of time that a cycle of a pulse wave spends in the higher of its two voltage states.

RAM – See Random Access Memory.

Random Access Memory (RAM) – The set of hardware locations in a computer where programs and data are stored.

random number generator – A circuit within the computer that generates pseudo-random numbers.

record – A collection of information consisting of one or more related items.

recursion – A seemingly circular process in which a procedure calls itself repeatedly.

release – Rate at which a signal falls from sustain level to zero level on a traditional ADSR envelope generator.

respiration – The process of yielding energy from oxidative reactions in living cells.

resultant – The complex periodic waveform resulting from the addition of two or more sine wave components.

ribosome – An organelle consisting of protein and ribonucleic acid (RNA) that catalyzes the construction of proteins.

right ascension – The east/west coordinate of a celestial body along the celestial equator.

ring modulation – Mathematically combining two frequencies by outputting their sum and difference and suppressing the original frequencies.

sector – On magnetic storage medium, a section of a track constituting a unit of data storage.

SID - Sound Interface Device chip in the Commodore 64 computer.

sine wave – A single frequency oscillation that produces a pure fundamental waveform without harmonics.

statement separator – A character that separates BASIC statements.

string – A consecutive set of similar data items—usually bits or characters.

subroutine – A program segment or module that performs a specified function (e.g., debugging).

sustain – The level of a signal after attack and decay, and before release on a traditional ADSR envelope generator.

symbiotic – Two or more different organisms in a close association that is of mutual benefit.

synchronization – The linking of two oscillators such that the start one oscillator cycle triggers the start of the second oscillator's cycle. One oscillator acts as the master and the other acts as a slave.

text file – A file containing textual information that is stored in ASCII format.

tile – A technique employed in conjunction with the IBM BASIC **PAINT** command that allows the user to place a specified graphic pattern onto a particular area of the screen.

TOD – An acronym for the Time-Of-Day clock on the Commodore 64 computer.

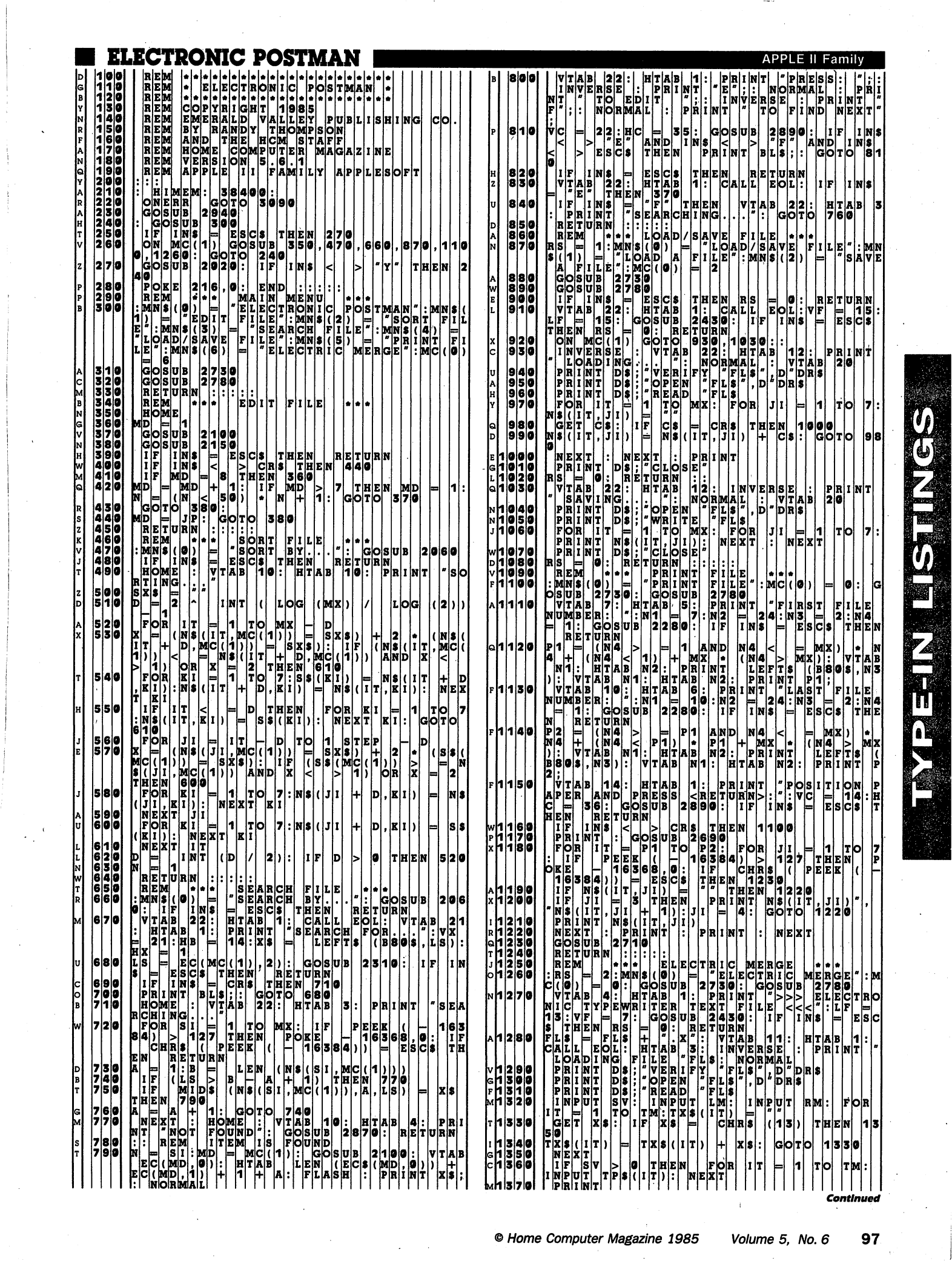
video buffer – A section in memory reserved for video processing.

waveform – 1. A graphic representation of the shape of a wave. 2. A signal with periodic fluctuations produced by an oscillator.

weight – The value of a bit determined by its position.

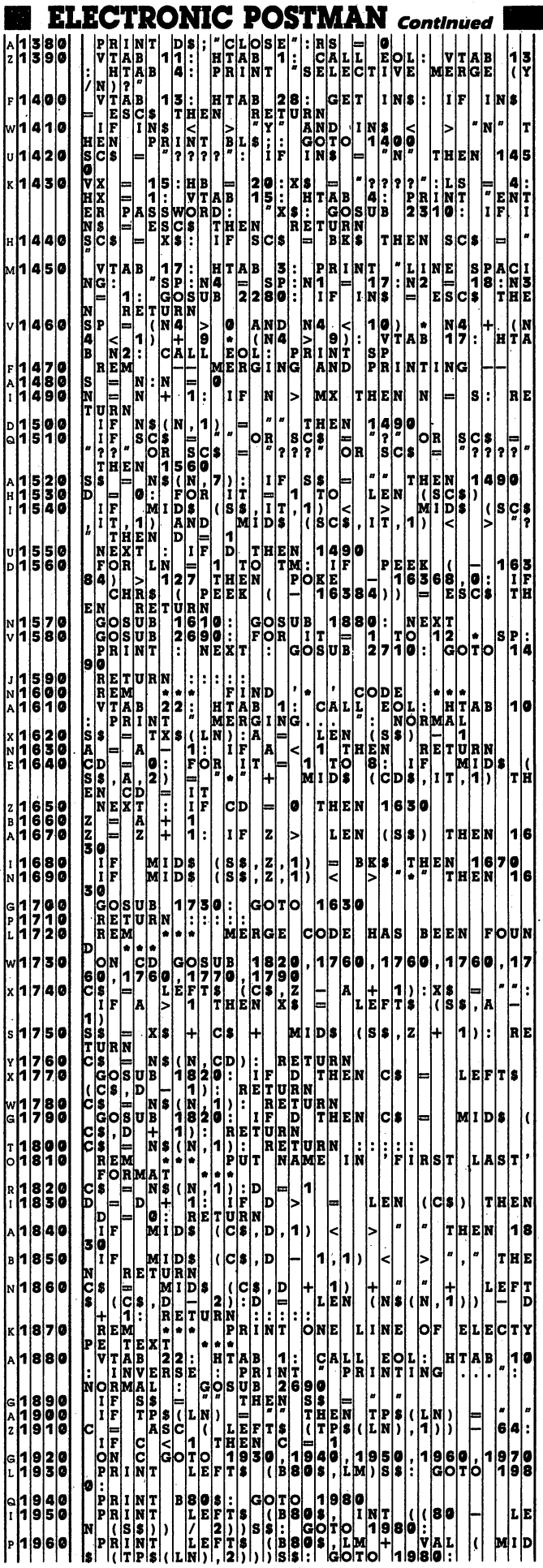
white noise – Random acoustical or electrical signals in which the intensity is the same at all frequencies within a given band.

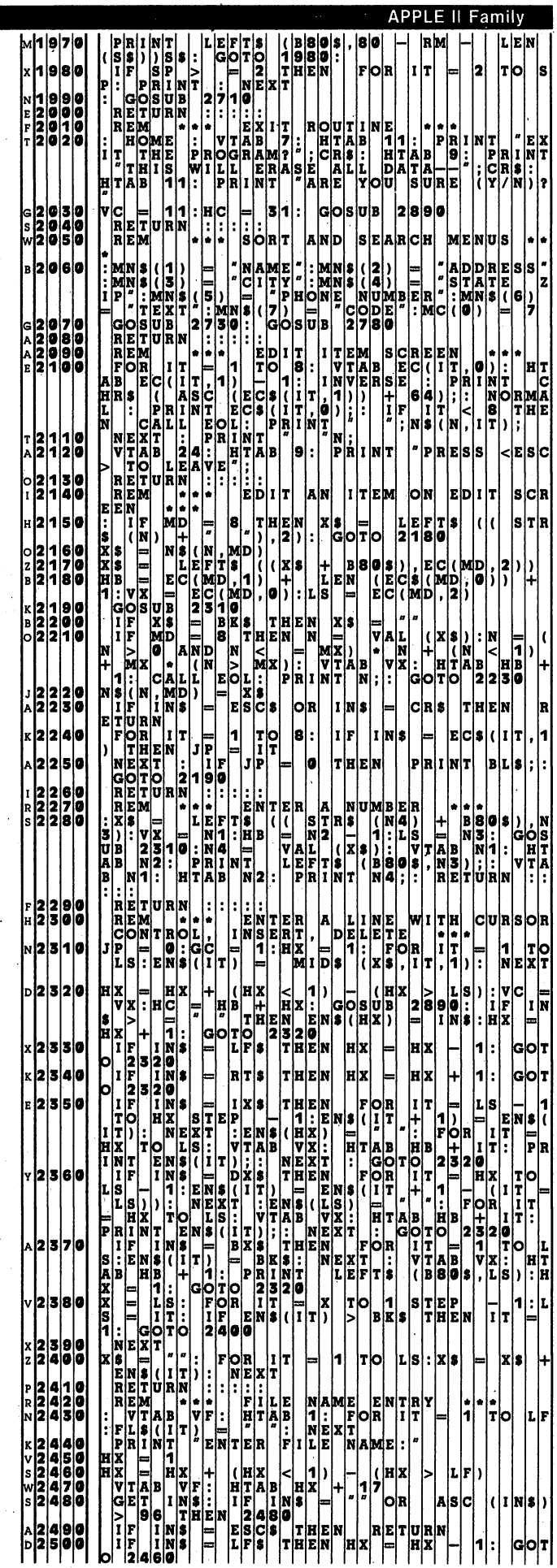
zodiac – An imaginary belt in the sky about 18 degrees wide along which the sun and all planets except Pluto appear to travel.



Continued

ELECTRONIC POSTMAN *Continued*

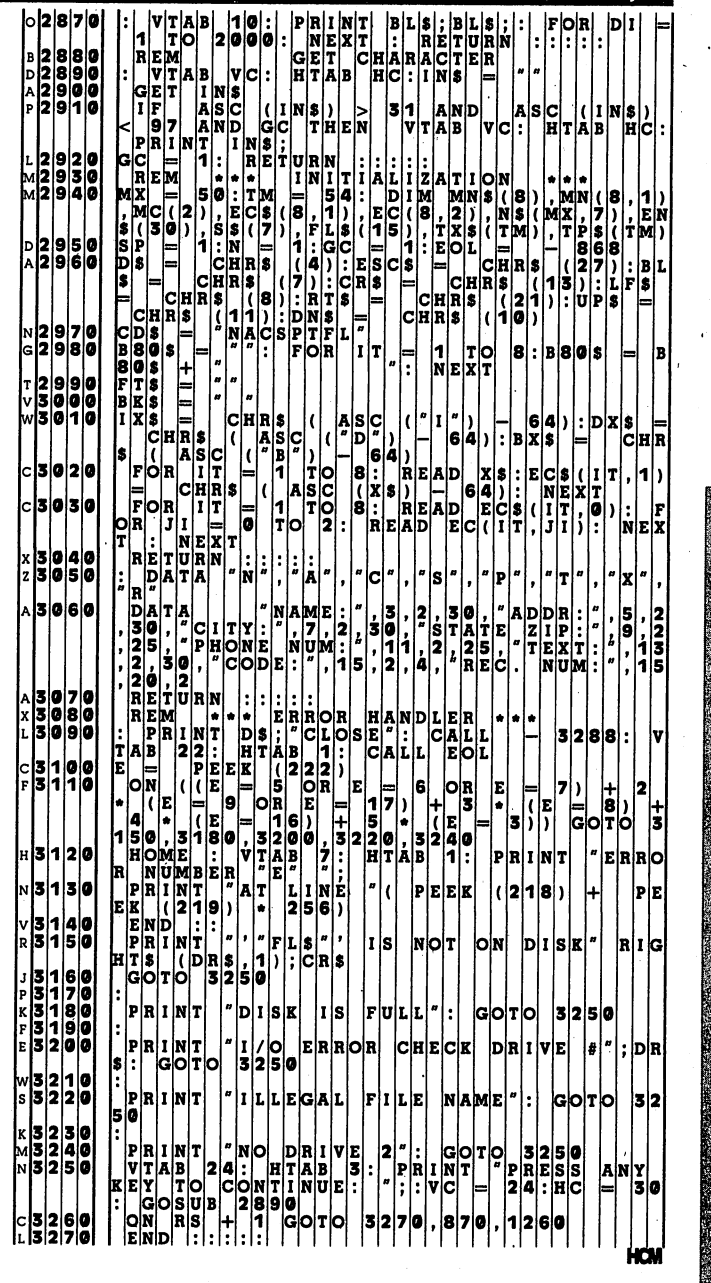




Continued

ELECTRONIC POSTMAN *Continued*

APPLE II Family

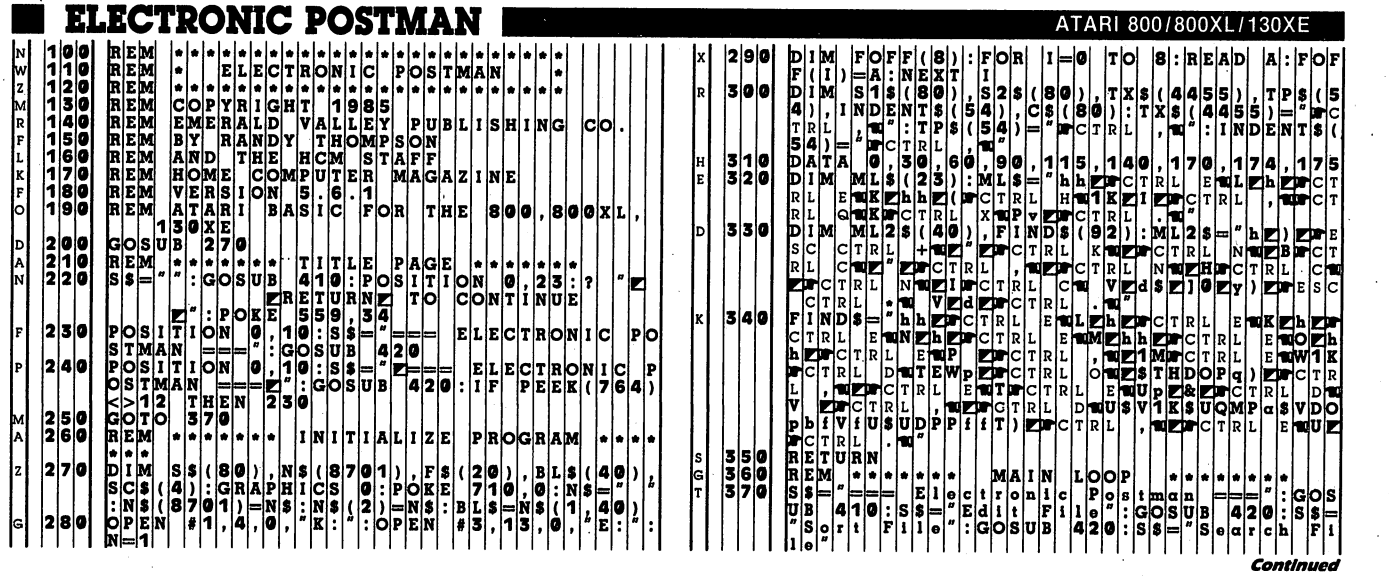


HCM

TYPE-IN LISTINGS

ELECTRONIC POSTMAN

ATARI 800/800XL/130XE



Continued



Continued

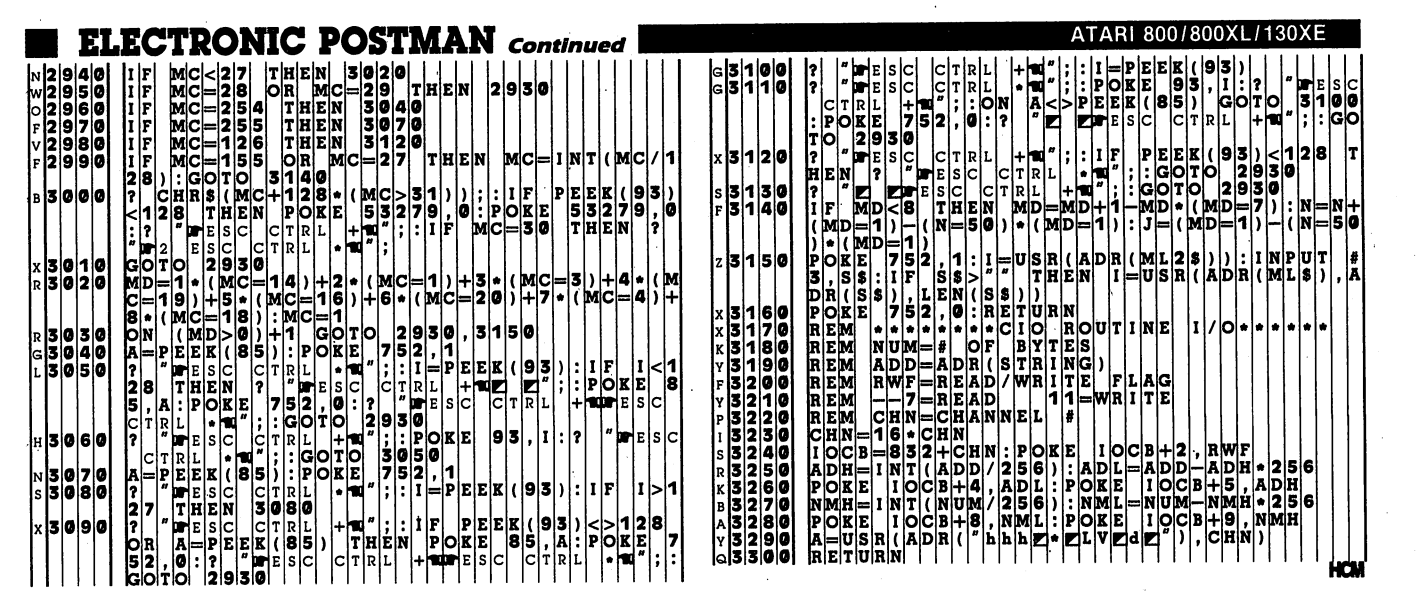


Continued

TYPE-IN LISTINGS

ELECTRONIC POSTMAN *Continued*

ATARI 800/800XL/130XE



HCM

ELECTRONIC POSTMAN

COMMODORE 64



Continued

```
10 A=A-1:GOTO970
20 NEXT:S$="NOT FOUND":GOSUB3300:RETURN
30 REM ITEM$ IS FOUND
40 N=PRINT#KS(MC,I):X=12:Y=15:GOSUB3410
50 :IF D=0 THEN GOSUB2750:D=1
60 N$(MC,I)=KS$,LEN(KS$)+1,L)+CTRL RVSON
70 +MID$(KS$,LEN(KS$)+1,L)+CTRL RVSOFF
80 +RIGHT$(KS$,L-A)
90 GOSUB2650:N$(MC,I)=KS$:POKE 781,22:S
100 YS=99:SYS 99:POKE 781,24:SYS 99
110 X=16:Y=21:GOSUB3410:PRINT"CMDR YEL
120 SHIFT"PRESS:X=1:Y=23:GOSUB3410
130 PRINT"CTRL WHT SHIFT RED CMDR YEL
140 F CMDR SHIFT YEL WHT CTRL RVS
150 ET"KS":IF KS<>"" AND KS<>"E" AND K
160 $="F"THEN1080
170 KS="E":GOSUB3300:RETURN
180 X=12:Y=15:GOSUB3410:PRINT"CTRL RVS
190 SHIFT/O"=====:GOSUB360:IF EX THEN RET
200 URN
210 POKE 781,6:SYS 59903:POKE 781,8:SYS
220 59903:PRINT"SHIFT L"OAD/"SHIFT
230 ":GOSUB3300:IF KS="" THEN 230
240 IF KS="E" THEN RETURN
250 X=24:GOSUB3410:PRINT"CTRL RVS
260 ON MC GOTO1126
270 ,24:SYS 59903:ON MC GOTO1126
280 ,16:POKE 49421,8:SYS 49152 SS:POKE 49421
290 ,14:POKE 49421,8:SYS 49152 SS:POKE 49421
300 ,12:POKE 49421,8:SYS 49152 SS:POKE 49421
310 ,10:POKE 49421,8:SYS 49152 SS:POKE 49421
320 ,8:POKE 49421,8:SYS 49152 SS:POKE 49421
330 ,6:POKE 49421,8:SYS 49152 SS:POKE 49421
340 ,4:POKE 49421,8:SYS 49152 SS:POKE 49421
350 ,2:POKE 49421,8:SYS 49152 SS:POKE 49421
360 ,1:POKE 49421,8:SYS 49152 SS:POKE 49421
370 ,0:POKE 49421,8:SYS 49152 SS:POKE 49421
380 ,24:SYS 59903:ON MC GOTO1126
390 ,22:SYS 59903:ON MC GOTO1126
400 ,20:SYS 59903:ON MC GOTO1126
410 ,18:SYS 59903:ON MC GOTO1126
420 ,16:SYS 59903:ON MC GOTO1126
430 ,14:SYS 59903:ON MC GOTO1126
440 ,12:SYS 59903:ON MC GOTO1126
450 ,10:SYS 59903:ON MC GOTO1126
460 ,8:SYS 59903:ON MC GOTO1126
470 ,6:SYS 59903:ON MC GOTO1126
480 ,4:SYS 59903:ON MC GOTO1126
490 ,2:SYS 59903:ON MC GOTO1126
500 ,1:SYS 59903:ON MC GOTO1126
510 ,0:SYS 59903:ON MC GOTO1126
520 ,24:SYS 59903:ON MC GOTO1126
530 ,22:SYS 59903:ON MC GOTO1126
540 ,20:SYS 59903:ON MC GOTO1126
550 ,18:SYS 59903:ON MC GOTO1126
560 ,16:SYS 59903:ON MC GOTO1126
570 ,14:SYS 59903:ON MC GOTO1126
580 ,12:SYS 59903:ON MC GOTO1126
590 ,10:SYS 59903:ON MC GOTO1126
600 ,8:SYS 59903:ON MC GOTO1126
610 ,6:SYS 59903:ON MC GOTO1126
620 ,4:SYS 59903:ON MC GOTO1126
630 ,2:SYS 59903:ON MC GOTO1126
640 ,1:SYS 59903:ON MC GOTO1126
650 ,0:SYS 59903:ON MC GOTO1126
660 ,24:SYS 59903:ON MC GOTO1126
670 ,22:SYS 59903:ON MC GOTO1126
680 ,20:SYS 59903:ON MC GOTO1126
690 ,18:SYS 59903:ON MC GOTO1126
700 ,16:SYS 59903:ON MC GOTO1126
710 ,14:SYS 59903:ON MC GOTO1126
720 ,12:SYS 59903:ON MC GOTO1126
730 ,10:SYS 59903:ON MC GOTO1126
740 ,8:SYS 59903:ON MC GOTO1126
750 ,6:SYS 59903:ON MC GOTO1126
760 ,4:SYS 59903:ON MC GOTO1126
770 ,2:SYS 59903:ON MC GOTO1126
780 ,1:SYS 59903:ON MC GOTO1126
790 ,0:SYS 59903:ON MC GOTO1126
800 ,24:SYS 59903:ON MC GOTO1126
810 ,22:SYS 59903:ON MC GOTO1126
820 ,20:SYS 59903:ON MC GOTO1126
830 ,18:SYS 59903:ON MC GOTO1126
840 ,16:SYS 59903:ON MC GOTO1126
850 ,14:SYS 59903:ON MC GOTO1126
860 ,12:SYS 59903:ON MC GOTO1126
870 ,10:SYS 59903:ON MC GOTO1126
880 ,8:SYS 59903:ON MC GOTO1126
890 ,6:SYS 59903:ON MC GOTO1126
900 ,4:SYS 59903:ON MC GOTO1126
910 ,2:SYS 59903:ON MC GOTO1126
920 ,1:SYS 59903:ON MC GOTO1126
930 ,0:SYS 59903:ON MC GOTO1126
940 ,24:SYS 59903:ON MC GOTO1126
950 ,22:SYS 59903:ON MC GOTO1126
960 ,20:SYS 59903:ON MC GOTO1126
970 ,18:SYS 59903:ON MC GOTO1126
980 ,16:SYS 59903:ON MC GOTO1126
990 ,14:SYS 59903:ON MC GOTO1126
1000 ,12:SYS 59903:ON MC GOTO1126
1010 ,10:SYS 59903:ON MC GOTO1126
1020 ,8:SYS 59903:ON MC GOTO1126
1030 ,6:SYS 59903:ON MC GOTO1126
1040 ,4:SYS 59903:ON MC GOTO1126
1050 ,2:SYS 59903:ON MC GOTO1126
1060 ,1:SYS 59903:ON MC GOTO1126
1070 ,0:SYS 59903:ON MC GOTO1126
1080 ,24:SYS 59903:ON MC GOTO1126
1090 ,22:SYS 59903:ON MC GOTO1126
1100 ,20:SYS 59903:ON MC GOTO1126
1110 ,18:SYS 59903:ON MC GOTO1126
1120 ,16:SYS 59903:ON MC GOTO1126
1130 ,14:SYS 59903:ON MC GOTO1126
1140 ,12:SYS 59903:ON MC GOTO1126
1150 ,10:SYS 59903:ON MC GOTO1126
1160 ,8:SYS 59903:ON MC GOTO1126
1170 ,6:SYS 59903:ON MC GOTO1126
1180 ,4:SYS 59903:ON MC GOTO1126
1190 ,2:SYS 59903:ON MC GOTO1126
1200 ,1:SYS 59903:ON MC GOTO1126
1210 ,0:SYS 59903:ON MC GOTO1126
1220 ,24:SYS 59903:ON MC GOTO1126
1230 ,22:SYS 59903:ON MC GOTO1126
1240 ,20:SYS 59903:ON MC GOTO1126
1250 ,18:SYS 59903:ON MC GOTO1126
1260 ,16:SYS 59903:ON MC GOTO1126
1270 ,14:SYS 59903:ON MC GOTO1126
1280 ,12:SYS 59903:ON MC GOTO1126
1290 ,10:SYS 59903:ON MC GOTO1126
1300 ,8:SYS 59903:ON MC GOTO1126
1310 ,6:SYS 59903:ON MC GOTO1126
1320 ,4:SYS 59903:ON MC GOTO1126
1330 ,2:SYS 59903:ON MC GOTO1126
1340 ,1:SYS 59903:ON MC GOTO1126
1350 ,0:SYS 59903:ON MC GOTO1126
1360 ,24:SYS 59903:ON MC GOTO1126
1370 ,22:SYS 59903:ON MC GOTO1126
1380 ,20:SYS 59903:ON MC GOTO1126
1390 ,18:SYS 59903:ON MC GOTO1126
1400 ,16:SYS 59903:ON MC GOTO1126
1410 ,14:SYS 59903:ON MC GOTO1126
1420 ,12:SYS 59903:ON MC GOTO1126
1430 ,10:SYS 59903:ON MC GOTO1126
1440 ,8:SYS 59903:ON MC GOTO1126
1450 ,6:SYS 59903:ON MC GOTO1126
1460 ,4:SYS 59903:ON MC GOTO1126
1470 ,2:SYS 59903:ON MC GOTO1126
1480 ,1:SYS 59903:ON MC GOTO1126
1490 ,0:SYS 59903:ON MC GOTO1126
1500 ,24:SYS 59903:ON MC GOTO1126
1510 ,22:SYS 59903:ON MC GOTO1126
1520 ,20:SYS 59903:ON MC GOTO1126
1530 ,18:SYS 59903:ON MC GOTO1126
1540 ,16:SYS 59903:ON MC GOTO1126
1550 ,14:SYS 59903:ON MC GOTO1126
1560 ,12:SYS 59903:ON MC GOTO1126
1570 ,10:SYS 59903:ON MC GOTO1126
1580 ,8:SYS 59903:ON MC GOTO1126
1590 ,6:SYS 59903:ON MC GOTO1126
1600 ,4:SYS 59903:ON MC GOTO1126
1610 ,2:SYS 59903:ON MC GOTO1126
1620 ,1:SYS 59903:ON MC GOTO1126
1630 ,0:SYS 59903:ON MC GOTO1126
1640 ,24:SYS 59903:ON MC GOTO1126
1650 ,22:SYS 59903:ON MC GOTO1126
1660 ,20:SYS 59903:ON MC GOTO1126
1670 ,18:SYS 59903:ON MC GOTO1126
1680 ,16:SYS 59903:ON MC GOTO1126
1690 ,14:SYS 59903:ON MC GOTO1126
1700 ,12:SYS 59903:ON MC GOTO1126
1710 ,10:SYS 59903:ON MC GOTO1126
1720 ,8:SYS 59903:ON MC GOTO1126
1730 ,6:SYS 59903:ON MC GOTO1126
1740 ,4:SYS 59903:ON MC GOTO1126
1750 ,2:SYS 59903:ON MC GOTO1126
1760 ,1:SYS 59903:ON MC GOTO1126
1770 ,0:SYS 59903:ON MC GOTO1126
1780 ,24:SYS 59903:ON MC GOTO1126
1790 ,22:SYS 59903:ON MC GOTO1126
1800 ,20:SYS 59903:ON MC GOTO1126
1810 ,18:SYS 59903:ON MC GOTO1126
1820 ,16:SYS 59903:ON MC GOTO1126
1830 ,14:SYS 59903:ON MC GOTO1126
1840 ,12:SYS 59903:ON MC GOTO1126
1850 ,10:SYS 59903:ON MC GOTO1126
1860 ,8:SYS 59903:ON MC GOTO1126
1870 ,6:SYS 59903:ON MC GOTO1126
1880 ,4:SYS 59903:ON MC GOTO1126
1890 ,2:SYS 59903:ON MC GOTO1126
1900 ,1:SYS 59903:ON MC GOTO1126
1910 ,0:SYS 59903:ON MC GOTO1126
1920 ,24:SYS 59903:ON MC GOTO1126
1930 ,22:SYS 59903:ON MC GOTO1126
1940 ,20:SYS 59903:ON MC GOTO1126
1950 ,18:SYS 59903:ON MC GOTO1126
1960 ,16:SYS 59903:ON MC GOTO1126
1970 ,14:SYS 59903:ON MC GOTO1126
1980 ,12:SYS 59903:ON MC GOTO1126
1990 ,10:SYS 59903:ON MC GOTO1126
2000 ,8:SYS 59903:ON MC GOTO1126
2010 ,6:SYS 59903:ON MC GOTO1126
2020 ,4:SYS 59903:ON MC GOTO1126
2030 ,2:SYS 59903:ON MC GOTO1126
2040 ,1:SYS 59903:ON MC GOTO1126
2050 ,0:SYS 59903:ON MC GOTO1126
2060 ,24:SYS 59903:ON MC GOTO1126
2070 ,22:SYS 59903:ON MC GOTO1126
2080 ,20:SYS 59903:ON MC GOTO1126
2090 ,18:SYS 59903:ON MC GOTO1126
2100 ,16:SYS 59903:ON MC GOTO1126
2110 ,14:SYS 59903:ON MC GOTO1126
2120 ,12:SYS 59903:ON MC GOTO1126
2130 ,10:SYS 59903:ON MC GOTO1126
2140 ,8:SYS 59903:ON MC GOTO1126
2150 ,6:SYS 59903:ON MC GOTO1126
2160 ,4:SYS 59903:ON MC GOTO1126
2170 ,2:SYS 59903:ON MC GOTO1126
2180 ,1:SYS 59903:ON MC GOTO1126
```

Continued

TYPE-IN LISTINGS

TYPE-IN LISTINGS

```
B 21900 IF MID$(S$,Z,1)<>"•" THEN 2130
A 22000 GOSUB 2210:GOTO 2130 FOUND--MERGE CODE
L 22100 REM CODE HAS BEEN FOUND--MERGE CODE
X 22200 ON CD GOSUB 2290,2240,2240,2240,2240
  2240,2240,2250,2270
X 22300 C$=LEFT$(S$,A-1):S$=LEFT$(S$,Z-A+1):S$=C$+S$:RETURN
C 22400 C$=MID$(CD,N):RETURN
H 22500 GOSUB 2290:IF D THEN C$=LEFT$(C$,D-1):RETURN
I 22600 GOSUB 2290:IF D THEN C$=MID$(C$,D+1):RETURN
I 22800 C$=NP$(T,N):RETURN
Z 22900 REM NP$(T,N) IN "FIRST LAST" FORMAT
F 23000 C$=N$(T,N):D=LEN(C$):THEN D=0:RETURN
O 23100 D=D+1:IF MID$(C$,D,1)<>" " THEN 2310
W 23200 IF MID$(C$,D,1)<>"<" THEN RETURN
Z 23300 C$=MID$(C$,D)+1):D=D+1:RETURN
J 23400 C$=LEFT$(C$,D-2):D=LEN(N$):ONE TEXT
I 23500 REM ? 4,T P$(LN),1,1):IF C$="" THEN
N 23600 POKE 4,7:X=INT(.. 15:GOSUB 3410:PRINT
R 23700 C$="MID$(T P$(LN),1,1):IF C$="" THEN
  C$="ASC(C$)-64+128*(ASC(C$)>70) GOSUB
Z 23800 ON 2400,2410,2420,2430,2440
F 23900 FOR I=1 TO SP:PRINT#4:NEXT:CLOSE 4:
  RETURN
T 24000 PRINT#4,LEFT$(BL$,LM)S$;
T 24100 PRINT#4,LEFT$(BL$,INT((80-LEN(S$))/
S 24200 2))INT$;:RETURN
A 24300 PRINT$(BL$,LM+VAL(MID$(TP$(L
N),INT$)),LEFT$(BL$,80-RM-LEN(S$))S$;
S 24400 PRINT#4,S$;:RETURN
L 24500 REM FILE TO LOAD ELECTRONIC TYPEWRITER TEXT
H 24600 POKE 781,10:SYS 59903:X=14:Y=15:GOS
  UB 3410:PRINT"CMDR YEL SHIFT L OAD
U 24700 IF $$="D" THEN 2490
U 24800 OPEN 1,8,0,F$+".X":GOTO 2500
S 24900 OPEN 1,8,8,F$+".X":GOSUB 3270
  :IF E THEN 2590
L 25000 INPUT#1,J
I 25100 INPUT#1,LM
N 25200 INPUT#1,RM
Y 25300 INPUT#1,TO 54:TX$(I)="
G 25400 INPUT#1,S$:IF S$="CTRL BLK" THEN S$
  =CHR$(13) THEN 2570
K 25500 TX$(I)=TX$(I)+S$:GOTO 2540
K 25600
T 25700 NEXT:I
N 25800 IF S$="D" THEN GOSUB 2610
C 25900 CLOSE 1:CLOSE 15:RETURN
S 26000 REM INPUT#1,TP$(54)
I 26100 FOR I=1 TO SP:INPUT#1,TP$(I):IF TP$(I)="CTRL BLK
Z 26200 NEXT:INPUT#1,TP$(I)="
X 26400 REM EDITING SCREEN
N 26600 PRINT"CTRL WHT":X=8:Y=1:GOSUB 3410:PR
  INT LEFT$(BL$,30)):GOSUB 3410:PRINT
P 26700 Y=4:GOSUB 3410:PRINT LEFT$(BL$,30):G
I 26800 OSUB 3410:PRINT N$(2,N)
Y 26900 Y=5:GOSUB 3410:PRINT LEFT$(BL$,30):G
X 27000 OSUB 3410:PRINT N$(3,N)
J 27100 X=13:Y=10:GOSUB 3410:PRINT LEFT$(BL$
  ,26):GOSUB 3410:PRINT N$(4,N)
B 27200 Y=13:GOSUB 3410:PRINT LEFT$(BL$,
  30):GOSUB 3410:PRINT N$(5,N)
U 27300 Y=20:GOSUB 3410:PRINT LEFT$(BL$
  ,5):GOSUB 3410:PRINT N;
R 27400 RETURN
R 27500 REM DRAW EDITING SCREEN
E 27600 PRINT"SHIFT CLR CTRL RVSON CTRL
  YEL 3 CMDR SHIFT C CMDR 12 SHIFT C
S 27700 CMDR 19 SHIFT C CMDR
F 27800 PRINT"CTRL RVSON SHIFT F 1 CTRL
  RVSOFF CMDR YEL SHIFT N AME CTRL
P 27900 PRINT" SHIFT B "
Q 28000 PRINT"CTRL RVSON 3 CMDR O CTRL R
  VSOFF 4 SHIFT C CMDR E 12 SHIFT C
U 28100 PRINT"19 SHIFT C CMDR X "
A 28200 PRINT"CTRL RVSON 3 CMDR U CTRL R
  VSOFF 4 SHIFT C CMDR R 5 SHIFT C
H 28300 PRINT"19 SHIFT C CMDR S "
N 28400 PRINT"CTRL RVSON SHIFT F 2 CTRL
  RVSOFF CMDR YEL SHIFT A DDR CTRL
C 28500 PRINT"
```

```
A 28600 PRINT"CTRL RVSON 3 CMDR O CTRL R
  VSOFF 4 SHIFT C CMDR E 12 SHIFT C
K 28700 PRINT"19 SHIFT C CMDR X ";
O 28800 PRINT"CTRL RVSON 3 CMDR U CTRL R
  VSOFF 4 SHIFT C CMDR R 12 SHIFT C
X 28900 PRINT"19 SHIFT C CMDR S ";
Z 29000 PRINT"CTRL RVSON SHIFT F 3 CTRL
  RVSOFF CMDR YEL SHIFT C ITY CTRL
  YEL SHIFT - ";
T 29100 PRINT" SHIFT B "
F 29200 PRINT"CTRL RVSON 3 CMDR O CTRL R
  VSOFF 4 SHIFT C CMDR E 12 SHIFT C
H 29300 PRINT"19 SHIFT C CMDR X ";
J 29400 PRINT"CTRL RVSON 3 CMDR U CTRL R
  VSOFF 9 SHIFT C CMDR R 7 SHIFT C
U 29500 PRINT"19 SHIFT C CMDR S ";
X 29600 PRINT"CTRL RVSON SHIFT F 4 CTRL
  RVSOFF CMDR YEL SHIFT S TATE SHIS
  HIFT - SHIFT P CTRL YEL SHIFT
T 29700 PRINT" SHIFT B "
Z 29800 PRINT"CTRL RVSON 3 CMDR O CTRL R
  VSOFF 9 SHIFT C CMDR E 7 SHIFT C
V 29900 PRINT"19 SHIFT C CMDR X ";
R 30000 PRINT"CTRL RVSON 3 CMDR U CTRL R
  VSOFF 9 SHIFT C CMDR R 7 SHIFT C
W 30100 PRINT"19 SHIFT C CMDR S ";
P 30200 PRINT"CTRL RVSON SHIFT F 5 CTRL
  RVSOFF CMDR YEL SHIFT P HONE SHI
  FT - SHIFT
T 30300 PRINT" SHIFT B "
X 30400 PRINT"CTRL RVSON 3 CMDR O CTRL R
  VSOFF 9 SHIFT C CMDR E 7 SHIFT C
N 30500 PRINT"19 SHIFT C CMDR X ";
V 30600 PRINT"
M 30700 PRINT"
C 30800 PRINT"CTRL RVSON CMDR GRN 3 CMDR
  UCTRL RVSOFF 4 SHIFT C CMDR R
A 30900 PRINT"19 SHIFT C CMDR S ";
A 31000 PRINT"CTRL RVSON SHIFT F 6 CTRL
  RVSOFF CMDR YEL SHIFT T EXT CMDR
  GRN SHIFT - ";
Y 31100 PRINT" SHIFT B "
I 31200 PRINT"CTRL RVSON 3 CMDR O CTRL R
  VSOFF 4 SHIFT C CMDR E 12 SHIFT C
S 31300 PRINT"19 SHIFT C CMDR X ";
S 31400 PRINT"CTRL RVSON 3 CMDR U CTRL R
  VSOFF 4 SHIFT C CMDR R 5 SHIFT C
R 31500 PRINT"CTRL RVSON 5 CMDR BLU 3
  CMDR RVSOFF 5 SHIFT CMDR YEL SH
  IFT F 8 CTRL RVSOFF CMDR BLU SHIFT -
N 31600 PRINT"CTRL RVSON CMDR GRN 3 CMDR
  RVSOFF 4 SHIFT C CMDR E
N 31700 PRINT"CTRL RVSON CMDR BLU SH
  IFT - SHIFT R EC CMDR
I 31800 PRINT"CTRL RVSON CMDR GRN 3 CMDR
  RVSOFF 4 SHIFT C CMDR E
I 31900 PRINT"CTRL RVSON CMDR BLU 3
  CMDR RVSOFF 5 SHIFT CMDR YEL SH
  IFT - SHIFT
U 32000 PRINT"CTRL BLU 10 CMDR
R 32100 PRINT"7 CMDR P
I 32200 PRINT"CMDR L CTRL RVSON
  CTRL WHT SHIFT P RESS - T ";
Z 32300 PRINT"O EXIT CTRL RVSOFF CTRL BLU
J 32400 PRINT"10 CMDR Y ";
P 32500 PRINT"7 CMDR Y CTRL WHT HOME ";
N 32600 RETURN
F 32700 REM READ DISK ERROR
E 32800 OPEN 15,8,15:INPUT#15,E,E$:IF E THE
  N E$="":GOSUB 3300
A 32900 RETURN
G 33000 REM BUZZ AND PAUSE
B 33100 X=INT((32-LEN(S$))/2):Y=15:GOSUB 341
  0:PRINT"YEL CMDR • ";S$;" • ";
J 33200 FOR I=1 TO 25:POKE 54296,15:POKE 54
  296,0:NEXT:FOR I=1 TO 2000:NEXT
V 33300 GOSUB 3410:PRINT" ";:RETURN
E 33400 REM EXIT PROGRAM
F 33500 PRINT"SHIFT CLR 7 CRSR RDOWN CTRL
  RVSON SHIFT E XIT THE SHI
  FT - SHIFT R OGRAM? CRSR RDOWN
P 33600 PRINT" SHIFT T HIS WILL ERASE
  ALL DATA. CRSR RDOWN"
```

Continued

ELECTRONIC POSTMAN *Continued*

COMMODORE 64

```
A 3370 PRINT"      {SHIFT A}RE YOU SURE (Y/N)?"
A 3380 GET K$:IF K$<>"Y" AND K$<>"N" AND K$<>"←" THEN3380
T 3390 IF K$="N" OR K$="←" THEN RETURN
U 3400 PRINT"{SHIFT CLR}BYE...":POKE 650,0:POKE 53272,21:POKE 657,0:END
N 3410 REM PLOT CURSOR AT X,Y
D 3420 POKE 781,Y:POKE 782,X:POKE 783,0:SYS 65520:RETURN
C 3430 DATA 15,"       {SHIFT E}DIT {SHIFT F}ILE","       {SHIFT S}ORT {SHIFT F}ILE","   {SHIFT SPACE}   {SHIFT S}EARCH {SHIFT F}ILE"
R 3440 DATA "    {SHIFT L}OAD/{SHIFT S}AVE {SHIFT F}ILE","    {SHIFT P}RINT {SHIFT F}ILE","    {SHIFT E}LECTRIC {SHIFT M}ERGE"
M 3450 DATA "       {SHIFT N}AME","    {SHIFT A}DDRESS","    {SHIFT C}ITY"
X 3460 DATA "   {SHIFT S}TATE","{SHIFT Z}IP","   {SHIFT P}HONE {SHIFT N}UMBER","   {SHIFT T}EXT"
A 3470 DATA "     {SHIFT C}ODE","    {SHIFT L}OAD {SHIFT F}ILE","    {SHIFT S}AVE {SHIFT F}ILE"
K 3480 DATA 3,6,13,15
V 3490 REM READ MACHINE LANGUAGE ROUTINE
L 3500 A=0:FOR I=49152 TO 49440:READ D:A=A+D:POKE I,D:NEXT:IF A=39677 THEN RETURN
Q 3510 PRINT"{SHIFT CLR}***{CTRL RVS ON} KEY IN ERROR IN DATA STATEMENTS! {CTRL RVS OFF}***":END
V 3520 DATA 162,0,134,198,134,212,134,199
N 3530 DATA 24,165,209,101,211,133,251,198
R 3540 DATA 251,138,101,210,133,252,24,165
R 3550 DATA 211,109,11,193,133,213,162,23
Z 3560 DATA 181,217,9,128,149,217,202,16
E 3570 DATA 247,32,139,176,133,73,132,74
N 3580 DATA 32,159,192,162,0,221,22,193
G 3590 DATA 240,123,232,236,12,193,208,245
U 3600 DATA 162,0,221,14,193,240,233,232
Y 3610 DATA 236,13,193,208,245,172,8,193
W 3620 DATA 240,17,201,157,240,4,201,20
G 3630 DATA 208,9,32,241,192,206,8,193
G 3640 DATA 24,144,205,204,11,193,240,200
H 3650 DATA 201,29,240,43,201,148,208,22
T 3660 DATA 170,172,11,193,177,251,201,32
B 3670 DATA 208,182,138,238,8,193,32,241
C 3680 DATA 192,206,8,193,240,170,201,193
S 3690 DATA 144,3,45,7,193,205,9,193
C 3700 DATA 144,158,205,10,193,176,153,32
A 3710 DATA 241,192,238,8,193,208,145,169
A 3720 DATA 0,133,204,133,207,32,228,255
N 3730 DATA 201,0,240,243,201,160,208,2
A 3740 DATA 169,32,133,2,96,32,246,192
C 3750 DATA 230,204,169,0,141,8,193,168
D 3760 DATA 132,253,200,177,251,201,32,240
Z 3770 DATA 14,132,253,144,8,201,64,144
E 3780 DATA 6,9,128,208,2,9,64,153
H 3790 DATA 255,1,204,11,193,208,227,164
X 3800 DATA 253,200,169,0,153,255,1,170
O 3810 DATA 160,2,32,137,180,32,218,169
U 3820 DATA 96,165,2,32,210,255,172,8
H 3830 DATA 193,200,177,251,41,127,145,251
K 3840 DATA 169,0,133,216,133,212,96,127
Q 3850 DATA 0,32,94,20,2,1,0,34
Q 3860 DATA 36,42,44,58,63,64,13,95
V 3870 DATA 133,137,134,138,135,139,136,140,147,0
```

HCM

ELECTRONIC POSTMAN

IBM PC/PCjr, TANDY 1000

```
W 100 ' ********************
P 110 ' * ELECTRONIC POSTMAN *
K 120 ' ********************
X 130 ' COPYRIGHT 1985
W 140 ' EMERALD VALLEY PUBLISHING CO
M 150 ' BY RANDY THOMPSON
Q 160 ' AND THE HCM STAFF
Z 170 ' HOME COMPUTER MAGAZINE
Q 180 ' VERSION 5.6.1
F 190 ' DOS 2.1 AND EITHER
        IBM PCjr W/ CARTRIDGE BASIC OR
M 200 ' IBM PC W/ BASICA OR
F 210 ' TANDY 1000 W/ GW BASIC
Y 220 KEY OFF:SCREEN 0:WIDTH 40:COLOR 15,1,8:CLS
V 230 T$ = " ELECTRONIC POSTMAN ":COLOR 8,2
A 240 LOCATE 4,10,0:PRINT STRING$(20," ");:LOCATE 6,10:PRINT STRING$(20," ");
N 250 LOCATE 5,10:PRINT T$;
B 260 X=15:Y=8:W=24:H=5:COLOR 12,3:GOSUB 1770
G 270 COLOR 15,12:LOCATE X+2,Y+2:PRINT"PRESS ";CHR$(17);CHR$(217);" TO CONTINUE";
D 280 K$ = INKEY$:IF K$ <> CHR$(13) THEN 280
W 290 'INITIALIZE PROGRAM
N 300 COLOR 15,1:CLS:LOCATE 10,4:COLOR 12,2:PRINT" PLEASE WAIT ";
B 310 READ D:DIM M$(D):FOR I = 1 TO D:READ M$(I):NEXT
M 320 READ D:DIM ME(D):FOR I = 1 TO D:READ ME(I):NEXT
C 330 READ D:DIM L(D,2):FOR I=1 TO D:READ L(I,1),L(I,2):NEXT
T 340 FOR D = 1 TO 10:KEY D, CHR$(200+D):NEXT
G 350 MX = 50:DIM N$(7,MX), TX$(54), TP$(54):N = 1
H 360 BL$ = STRING$(80, " "):CD$ = "NACSPTFL"
W 370 'MAIN LOOP
N 380 M = 1:S$ = "    ===  "+T$+"  ===":GOSUB 400:IF EX THEN GOSUB 1500:GOTO 380
X 390 ON MC GOSUB 570, 1400, 780, 990,1190, 2670:GOTO 380
W 400 'GENERAL PURPOSE MENU ROUTINE
Z 410 COLOR 15,1:CLS:LOCATE 25,1:PRINT STRING$(40," ");:LOCATE 1,1:PRINT S$:EX=0:MC=1
P 420 LOCATE 22,1,0:PRINT STRING$(40,CHR$(196));
Y 430 LOCATE 23,1:COLOR 5:PRINT"USE CURSOR KEYS TO SELECT OPTION ";
W 440 LOCATE 24,1:PRINT"    PRESS [ESC] TO EXIT ";
J 450 D=3:FOR I=ME(M-1)+1 TO ME(M):COLOR 15,1:D=D+2:IF MC=I-ME(M-1) THEN COLOR 1,7:LOCATE D,10:PRINT ">" M$(I) "<";:GOTO 470
K 460 LOCATE D,10:PRINT " " M$(I) " ";
A 470 NEXT
R 480 K$ = INKEY$:IF K$ <> CHR$(27) AND K$ <> CHR$(13) AND K$ <> CHR$(0)+CHR$(72) AND K$ <> CHR$(0)+CHR$(80) THEN 480
B 490 IF K$ = CHR$(27) THEN EX = 1:RETURN
H 500 IF K$ = CHR$(13) THEN RETURN
N 510 IF K$ <> CHR$(0) + CHR$(80) THEN 540
H 520 IF MC < ME(M)-ME(M-1) THEN MC=MC+1:GOTO 450
E 530 MC = 1:GOTO 450
Z 540 IF K$ <> CHR$(0)+CHR$(72) THEN 450
M 550 IF MC > 1 THEN MC = MC - 1:GOTO 450
X 560 MC = ME(M) - ME(M-1):GOTO 450
D 570 'EDIT FILE
E 580 MD = 1:GOSUB 1580
X 590 GOSUB 1700:COLOR 15,0
Z 600 FC$ = " ":LC$ = CHR$(126)
L 610 ON MD GOSUB 680,680,680,690,690,700,710,720
N 620 IF K$=CHR$(27) THEN RETURN
C 630 IF K$=CHR$(210) THEN LOCATE X,Y:PRINT STRING$(LEN(N$(MD,N))," ");:N$(MD,N)="":GOTO 610
M 640 IF K$ = CHR$(13) THEN 660
H 650 IF K$ >= CHR$(201) AND K$ <= CHR$(208) THEN MD = ASC(K$)-200:GOTO 610
F 660 IF MD=8 THEN 610 ELSE MD=MD+1:IF MD = 8 THEN MD = 1:N = N - (N<MX):GOTO 590
D 670 GOTO 610
P 680 X=3*MD-1:Y=11:L=29:LOCATE X,Y:S$=N$(MD,N):GOSUB 1880:N$(MD,N)=S$:RETURN
S 690 X=3*MD-1:Y=16:L=24:LOCATE X,Y:S$=N$(MD,N):GOSUB 1880:N$(MD,N)=S$:RETURN
S 700 X=17:Y=11:L=29:LOCATE X,Y:S$ = N$(MD,N):GOSUB 1880:N$(MD,N) = S$:RETURN
P 710 X=21:Y=11:L=4:LOCATE X,Y:S$=N$(MD,N):GOSUB 1880:N$(MD,N)=S$:RETURN
X 720 X=21:Y=37:L=2:LOCATE X,Y:S$=MID$(STR$(N),2):GOSUB 1880
Z 730 IF K$ = CHR$(210) THEN GOTO 720
G 740 IF VAL(S$) > MX THEN N = MX:GOTO 760 ELSE IF VAL(S$) < 1 THEN N = 1:GOTO 760
W 750 N = VAL(S$)
R 760 GOSUB 1700
Q 770 RETURN
A 780 'SEARCH FILE
R 790 D1=0:M=2:S$="     === SEARCH FILE ==="+CHR$(13)+"     SEARCH BY...":GOSUB 400
N 800 IF EX THEN RETURN
M 810 COLOR 15,1:CLS:COLOR 4,2:LOCATE 3,1:PRINT"SEARCH BY:";M$(MC+ME(M-1))
X 820 COLOR 15,1:LOCATE 6,1,1:PRINT"SEARCH";:LOCATE 7,1:PRINT"   FOR...";
Y 830 X=7:Y=10:L=30:LC$=CHR$(126):FC$=" ":S$ = "":COLOR 14:GOSUB 1880
M 840 IF S$ = "" OR K$ = CHR$(27) THEN RETURN
```

Continued

TYPE-IN LISTINGS

```
10  '  ELECTRONIC POSTMAN
20  '  IBM PC/PCjr, TANDY 1000
30  '  Copyright (c) 1985 Home Computer Magazine
40  '  Volume 5, No. 6
50  '
60  '  Type in and save this program.
70  '  Run it to create a data file.
80  '  Then add the following lines:
90  '
100 NEXT:STOP
110 '
120 '  Subroutine to print a message
130 '  and wait for a keypress.
140 '
150 PRINT:PRINT"Press any key to continue"
160 WHILE INKEY$="":WEND:RETURN
170 '
180 '  Subroutine to print a message
190 '  and wait for a keypress.
200 '
210 PRINT:PRINT"Press any key to continue"
220 WHILE INKEY$="":WEND:RETURN
230 '
240 '  Subroutine to print a message
250 '  and wait for a keypress.
260 '
270 PRINT:PRINT"Press any key to continue"
280 WHILE INKEY$="":WEND:RETURN
290 '
300 '  Subroutine to print a message
310 '  and wait for a keypress.
320 '
330 PRINT:PRINT"Press any key to continue"
340 WHILE INKEY$="":WEND:RETURN
350 '
360 '  Subroutine to print a message
370 '  and wait for a keypress.
380 '
390 PRINT:PRINT"Press any key to continue"
400 WHILE INKEY$="":WEND:RETURN
410 '
420 '  Subroutine to print a message
430 '  and wait for a keypress.
440 '
450 PRINT:PRINT"Press any key to continue"
460 WHILE INKEY$="":WEND:RETURN
470 '
480 '  Subroutine to print a message
490 '  and wait for a keypress.
500 '
510 PRINT:PRINT"Press any key to continue"
520 WHILE INKEY$="":WEND:RETURN
530 '
540 '  Subroutine to print a message
550 '  and wait for a keypress.
560 '
570 PRINT:PRINT"Press any key to continue"
580 WHILE INKEY$="":WEND:RETURN
590 '
600 '  Subroutine to print a message
610 '  and wait for a keypress.
620 '
630 PRINT:PRINT"Press any key to continue"
640 WHILE INKEY$="":WEND:RETURN
650 '
660 '  Subroutine to print a message
670 '  and wait for a keypress.
680 '
690 PRINT:PRINT"Press any key to continue"
700 WHILE INKEY$="":WEND:RETURN
710 '
720 '  Subroutine to print a message
730 '  and wait for a keypress.
740 '
750 PRINT:PRINT"Press any key to continue"
760 WHILE INKEY$="":WEND:RETURN
770 '
780 '  Subroutine to print a message
790 '  and wait for a keypress.
800 '
810 PRINT:PRINT"Press any key to continue"
820 WHILE INKEY$="":WEND:RETURN
830 '
840 '  Subroutine to print a message
850 '  and wait for a keypress.
860 '
870 PRINT:PRINT"Press any key to continue"
880 WHILE INKEY$="":WEND:RETURN
890 '
900 '  Subroutine to print a message
910 '  and wait for a keypress.
920 '
930 PRINT:PRINT"Press any key to continue"
940 WHILE INKEY$="":WEND:RETURN
950 '
960 '  Subroutine to print a message
970 '  and wait for a keypress.
980 '
990 '  Subroutine to print a message
1000 '  and wait for a keypress.
1010 '
1020 '  Subroutine to print a message
1030 '  and wait for a keypress.
1040 '
1050 '  Subroutine to print a message
1060 '  and wait for a keypress.
1070 '
1080 '  Subroutine to print a message
1090 '  and wait for a keypress.
1100 '
1110 '  Subroutine to print a message
1120 '  and wait for a keypress.
1130 '
1140 '  Subroutine to print a message
1150 '  and wait for a keypress.
1160 '
1170 '  Subroutine to print a message
1180 '  and wait for a keypress.
1190 '
1200 '  Subroutine to print a message
1210 '  and wait for a keypress.
1220 '
1230 '  Subroutine to print a message
1240 '  and wait for a keypress.
1250 '
1260 '  Subroutine to print a message
1270 '  and wait for a keypress.
1280 '
1290 '  Subroutine to print a message
1300 '  and wait for a keypress.
1310 '
1320 '  Subroutine to print a message
1330 '  and wait for a keypress.
1340 '
1350 '  Subroutine to print a message
1360 '  and wait for a keypress.
1370 '
1380 '  Subroutine to print a message
1390 '  and wait for a keypress.
1400 '
1410 '  Subroutine to print a message
1420 '  and wait for a keypress.
1430 '
1440 '  Subroutine to print a message
1450 '  and wait for a keypress.
1460 '
1470 '  Subroutine to print a message
1480 '  and wait for a keypress.
1490 '
1500 '  Subroutine to print a message
1510 '  and wait for a keypress.
1520 '
1530 '  Subroutine to print a message
1540 '  and wait for a keypress.
1550 '
1560 '  Subroutine to print a message
1570 '  and wait for a keypress.
1580 '
1590 '  Subroutine to print a message
1600 '  and wait for a keypress.
1610 '
1620 '  Subroutine to print a message
1630 '  and wait for a keypress.
1640 '
1650 '  Subroutine to print a message
1660 '  and wait for a keypress.
1670 '
1680 '  Subroutine to print a message
1690 '  and wait for a keypress.
1700 '
1710 '  Subroutine to print a message
1720 '  and wait for a keypress.
1730 '
1740 '  Subroutine to print a message
1750 '  and wait for a keypress.
1760 '
1770 '  Subroutine to print a message
1780 '  and wait for a keypress.
1790 '
1800 '  Subroutine to print a message
1810 '  and wait for a keypress.
1820 '
1830 '  Subroutine to print a message
1840 '  and wait for a keypress.
1850 '
1860 '  Subroutine to print a message
1870 '  and wait for a keypress.
1880 '
1890 '  Subroutine to print a message
1900 '  and wait for a keypress.
1910 '
1920 '  Subroutine to print a message
1930 '  and wait for a keypress.
1940 '
1950 '  Subroutine to print a message
1960 '  and wait for a keypress.
1970 '
1980 '  Subroutine to print a message
1990 '  and wait for a keypress.
2000 '
2010 '  Subroutine to print a message
2020 '  and wait for a keypress.
2030 '
```

Continued

```
L 2040 FOR I=1 TO LEN(A$):C$=MID$(A$,I,1):IF C$<"A" OR C$>"Z" THEN 2060
X 2050 B$=B$+CHR$(ASC(C$)+128):NEXT:A$=B$:B$="":FOR I=1 TO LEN(A$):C$=MID$(A$,I,1)
W 2060 IF C$<"A" OR C$>"Z" THEN B$=B$+C$:NEXT:A$=B$:RETURN
Q 2070 IF C$<"A" OR C$>"Z" THEN 2080 ELSE C$=CHR$(ASC(C$)-128):B$=B$+C$:NEXT:A$=B$:RETURN
F 2080 B$=B$+C$:NEXT:A$=B$:RETURN
Y 2090 '
W 2100 '
E 2110 '
D 2120 '
O 2130 '
M 2140 '
M 2150 '
C 2160 '
L 2170 '
N 2180 '
W 2190 '
H 2200 '
P 2210 '
H 2220 '
D 2230 '
S 2240 '
S 2250 '
A 2260 '
N 2270 '
A 2280 '
U 2290 '
B 2300 '
R 2310 '
K 2320 '
D 2330 '
B 2340 '
N 2350 '
V 2360 '
O 2370 '
R 2380 '
L 2390 '
U 2400 '
K 2410 '
E 2420 '
B 2430 '
I 2440 '
B 2450 '
A 2460 '
C 2470 '
O 2480 '
Z 2490 '
Y 2500 '
O 2510 '
V 2520 '
Z 2530 '
Z 2540 '
N 2550 '
H 2560 '
V 2570 '
A 2580 '
A 2590 '
Q 2600 '
A 2610 '
X 2620 '
Z 2630 '
X 2640 '
I 2650 '
C 2660 '
S 2670 '
N 2680 '
O 2690 '
J 2700 '
Y 2710 '
Z 2720 '
P 2730 '
M 2740 '
Z 2750 '
Q 2760 '
D 2770 '
K 2780 '
S 2790 '
P 2800 '
D 2810 '
C 2820 '
D 2830 '
Q 2840 '
G 2850 '
L 2860 '
O 2870 '
P 2880 '
A 2890 '
J 2900 '
A 2930 '
X 2940 '
W 2950 '
N 2960 '
F 2970 '
Q 2980 '
U 2990 '
U 3000 '
A 3010 '
O 3020 '
P 3030 '
N 3040 '
X 3050 '
S 3060 '
E 3070 '
F 3080 '
U 3090 '
U 3100 '
Y 3110 '
O 3120 '
M 3130 '
E 3140 '
P 3150 '
O 3160 '
A 3170 '
U 3180 '
I 3190 '
W 3200 '
P 3210 '
Y 3220 '
G 3230 '
C 3240 '
E 3250 '
X 3260 '
N 3270 '
A 3280 '
A 3290 '
C 3300 '
R 3310 '
K 3320 '
N 3330 '
Y 3340 '
A 3350 '
G 3360 '
```

TYPE-IN LISTINGS

HCM

CARD SHUFFLER


```
100 !!!!!!!! *** SHUFFLE ***
110 !!!!!!!! *** CARD ***
120 !!!!!!!! *** PROGRAM ***
130 !!!!!!!! *** BY ***
140 !!!!!!!! *** TOM ***
150 !!!!!!!! *** HOVER ***
160 !!!!!!!! *** 5.6.D. ***
170 !!!!!!!! *** 1985 ***
180 !!!!!!!! *** PUBLISHING ***
190 !!!!!!!! *** COMPANY ***
200 !!!!!!!! *** MAGAZINE ***
210 !!!!!!!! *** SCREEN ***
220 !!!!!!!! *** COLOR ***
230 !!!!!!!! *** ROUTINE ***
240 !!!!!!!! *** BY ***
250 !!!!!!!! *** TOM ***
260 !!!!!!!! *** HOVER ***
270 !!!!!!!! *** 1985 ***
280 !!!!!!!! *** 5.6.D. ***
290 !!!!!!!! *** PUBLISHING ***
300 !!!!!!!! *** COMPANY ***
310 !!!!!!!! *** MAGAZINE ***
320 !!!!!!!! *** SCREEN ***
330 !!!!!!!! *** COLOR ***
340 !!!!!!!! *** ROUTINE ***
350 !!!!!!!! *** BY ***
360 !!!!!!!! *** TOM ***
370 !!!!!!!! *** HOVER ***
380 !!!!!!!! *** 1985 ***
390 !!!!!!!! *** 5.6.D. ***
400 !!!!!!!! *** PUBLISHING ***
410 !!!!!!!! *** COMPANY ***
420 !!!!!!!! *** MAGAZINE ***
430 !!!!!!!! *** SCREEN ***
440 !!!!!!!! *** COLOR ***
450 !!!!!!!! *** ROUTINE ***
460 !!!!!!!! *** BY ***
470 !!!!!!!! *** TOM ***
480 !!!!!!!! *** HOVER ***
490 !!!!!!!! *** 1985 ***
500 !!!!!!!! *** 5.6.D. ***
510 !!!!!!!! *** PUBLISHING ***
520 !!!!!!!! *** COMPANY ***
530 !!!!!!!! *** MAGAZINE ***
540 !!!!!!!! *** SCREEN ***
550 !!!!!!!! *** COLOR ***
560 !!!!!!!! *** ROUTINE ***
570 !!!!!!!! *** BY ***
```

```
580 !!!!!!!! *** TOM ***
590 !!!!!!!! *** HOVER ***
600 !!!!!!!! *** 1985 ***
610 !!!!!!!! *** 5.6.D. ***
620 !!!!!!!! *** PUBLISHING ***
630 !!!!!!!! *** COMPANY ***
640 !!!!!!!! *** MAGAZINE ***
650 !!!!!!!! *** SCREEN ***
660 !!!!!!!! *** COLOR ***
670 !!!!!!!! *** ROUTINE ***
680 !!!!!!!! *** BY ***
690 !!!!!!!! *** TOM ***
700 !!!!!!!! *** HOVER ***
710 !!!!!!!! *** 1985 ***
720 !!!!!!!! *** 5.6.D. ***
730 !!!!!!!! *** PUBLISHING ***
740 !!!!!!!! *** COMPANY ***
750 !!!!!!!! *** MAGAZINE ***
760 !!!!!!!! *** SCREEN ***
770 !!!!!!!! *** COLOR ***
780 !!!!!!!! *** ROUTINE ***
790 !!!!!!!! *** BY ***
800 !!!!!!!! *** TOM ***
810 !!!!!!!! *** HOVER ***
820 !!!!!!!! *** 1985 ***
830 !!!!!!!! *** 5.6.D. ***
840 !!!!!!!! *** PUBLISHING ***
850 !!!!!!!! *** COMPANY ***
860 !!!!!!!! *** MAGAZINE ***
870 !!!!!!!! *** SCREEN ***
880 !!!!!!!! *** COLOR ***
890 !!!!!!!! *** ROUTINE ***
900 !!!!!!!! *** BY ***
910 !!!!!!!! *** TOM ***
920 !!!!!!!! *** HOVER ***
930 !!!!!!!! *** 1985 ***
940 !!!!!!!! *** 5.6.D. ***
950 !!!!!!!! *** PUBLISHING ***
960 !!!!!!!! *** COMPANY ***
970 !!!!!!!! *** MAGAZINE ***
980 !!!!!!!! *** SCREEN ***
990 !!!!!!!! *** COLOR ***
1000 !!!!!!!! *** ROUTINE ***
1010 !!!!!!!! *** BY ***
1020 !!!!!!!! *** TOM ***
1030 !!!!!!!! *** HOVER ***
```

Continued

CARD SHUFFLER *Continued*

TI-99/4A

```
1040 DISPLAY AT(12,1):"ENTER NAME OF IND
     EXEC,1):::GOSUB 1210:::DISPLAY AT(
     FOR I=1 TO 4:::DISPLAY AT(I*3+3,1):
     PEN #1:::FOR I=1 TO 4:::FOR J=1 TO
     #1,C$(I,J):::NEXT J,I:::CLOSE #1::
     LAY AT(I*3+3,1):"PLAYER "&STR$(I)&"
     ALL I:::NEXT I:::DISPLAY AT(24,1):"
     ::CALL KEY(0,K,S):::IF S=0 THEN 10
     ::RETURN
1050 SUB PRESS :::DISPLAY AT(24,1):"PR
     ::CALL KEY(0,K,S):::IF S=0 THEN 10
     ::RETURN
1060 SUB PUTIT(I,J,C$):::OPEN #1:"SCR
     ,SIZE,#I,J):::PRINT #1:C$;:::CLOSE
1070 SUB PRINT(I,J,C$):::DISPLAY AT(I,
1080 SUB PRI(I,J,C$):::PRINT #1:AT(I,J
1090 SUB NEXT(I,J):::J=J+1:::IF J<=13
     J=1:::I=I+1:::IF I>=4 THEN I=4
1100 SUB ALLTHE(I,J,C$):::FOR I=I TO 4
     J TO 13:::DISPLAY AT(I,J):C$:::NEX
     NEXT I:::RETURN
1110 SUB WANT(I,J,C$):::DISPLAY AT(24,
     ::CALL KEY(0,K,S):::IF S=0 THEN 11
     ::RETURN
1120 SUB ENTER(I,J,C$):::DISPLAY AT(24,
     ::ACCEPT AT(I,J)SIZE(-2):C$:::RETU
1130 SUB CHAR(I,J,C$):::CALL HCHAR(I,J
1140 SUB SAY(I,J,C$):::DISPLAY AT(I,J)
1150 SUB SUBIT(I,J,C$):::DISPLAY AT(I,J
1160 SUB CALL(I,J,C$):::CALL HCHAR(I,J
1170 SUB RETURN :::SUBEND
1180 SUB CALL(I,J,C$):::CALL HCHAR(I,J
     2,VCHAR(I,J,ASC(C$),3):::SUBEND
1190 SUB CALL(I,J,C$):::CALL HCHAR(I,J
     2,RCHAR(I,J,C$,3):::SUBEND
1200 SUB KEY(I,K,S):::CALL KEY(I,K,S):
     ::IF K<>13 THEN 1200
1210 SUB DEVICE&FILE(I,J):::PRINT #I:"
     ::SUBEND
1220 SUB ELSE :::IF SEG$(S$,4,1)="(" T
     ::RETURN 1260
1230 SUB URL(S$):::IF POS(S$,"://",1)<
     ::IF POS(S$,"/",4)<1 THEN S$=S$&"/
     ::IF SEG$(S$,LEN(S$),1)="/" THEN S
     ::RETURN S$
1240 SUB ERROR :::PRINT "ERROR":::SUBE
1250 SUB DEBUG :::PRINT "DEBUG":::SUBE
1260 SUB DISPLAY(I,J,C$):::DISPLAY AT(
1270 SUB PLAY(I,J,C$):::PLAY C$,I,J:::
1280 SUB DISPLAY(I,J,C$):::DISPLAY AT(
1290 SUB HCHAR(I,J,C$):::CALL HCHAR(I,
     2,VCHAR(I,J,ASC(C$),3):::SUBEND
```

SERF CITY

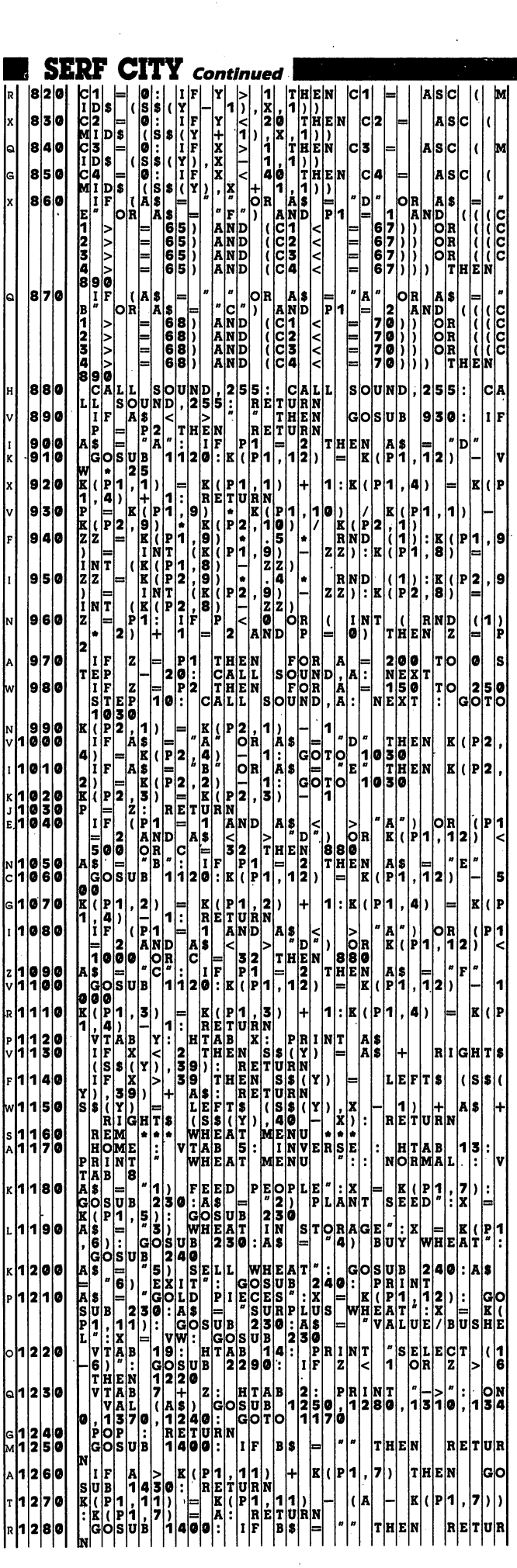
APPLE II Family

```
100  REM ********************
110  REM *  SERF CITY 1985  *
120  REM *  BY JOHN HANGIC  *
130  REM *  COPYRIGHT 1985  *
140  REM *  COMPUTE! BOOKS  *
150  REM *  ALL RIGHTS RESV *
160  REM *  WORLDWIDE       *
170  REM ********************
180  REM
190  REM
200  REM
210  REM
220  REM
230  BT$=CHR$(34):HOME:VTAB 12:PRINT "
     ";BT$;"SERF CITY";BT$;" IS LOADING
     ":PRINT "PLEASE WAIT"
240  HOME:VTAB 22:PRINT "SERF CITY IS L
     OADING"
250  REM
260  REM
270  REM
280  REM
290  REM
300  REM
310  REM
320  REM
330  REM
340  REM
350  REM
360  REM
370  REM
380  REM
390  REM
400  REM
410  REM
420  REM
430  REM
440  REM
450  REM
460  REM
470  REM
480  REM
490  REM
500  REM
510  REM
520  REM
530  REM
540  REM
550  REM
560  REM
570  REM
580  REM
590  REM
600  REM
610  REM
620  REM
630  REM
640  REM
650  REM
660  REM
670  REM
680  REM
690  REM
700  REM
710  REM
720  REM
730  REM
740  REM
750  REM
760  REM
770  REM
780  REM
790  REM
800  REM
810  REM
```

Continued

TYPE-IN LISTINGS

TYPE-IN LISTINGS



Continued



Continued

SERF CITY *Continued*

APPLE II Family

```
I 2740 IF Z = 11 THEN PRINT "INVALID FILENAME": GOTO 2770
M 2750 IF Z = 255 THEN TEXT : HOME : PRINT "PROGRAM ABORTED ON LINE #"D: END
W 2760 TEXT : HOME : PRINT "FATAL ERROR #"Z" IN LINE #"D: END
E 2770 FOR Z = 1 TO 2000: NEXT : VTAB 21: CALL - 958: IF P$(1) = "" THEN RUN
I 2780 GOTO 2310
A 2790 DATA12,2,1,9,0,0,0,90,0,0,500,500,0,0
K 2800 DATA"ACADFD","ABADED"
R 2810 DATA"ABADED","AAADDD"
Y 2820 DATA 9999
H 2830 DATA 76,133,8,76,139,8,32,76,231,160,32,138,202,208,253,44,48,192,170,136,208,246,96,160,0,36,50,48,2,160,255,132,6,169,7,133,7,162,170,164
O 2840 DATA 8,208,2,162,213,192,4,144,2,162,85,165,36,168,41,1,208,4,138,73,127,170,134,9,165,8,10,10,10,170,24,165,40,133,38,165,41,105,24,133
A 2850 DATA39,24,165,39,105,4,133,39,189,169,8,69,6,37,9,145,38,232,198,7,16,235,96,133,69,134,70,132,71,201,160,144,11,201,207,176,7,41,31,133
L 2860 DATA8,32,23,8,165,69,166,70,164,71,76,240,253,160,103,169,8,208,4,160,240,169,253,174,9,191,224,76,208,12,174,15,191,208,7,140,48,190,141,49
V 2870 DATA190,96,132,54,133,55,76,234,3,0,0,0,0,0,0,0,0,127,127,127,127,127,127,127,127,127,99,65,64,76,76,64,127,127,1,103,1,103,103,1
E 2880 DATA127,127,127,127,127,127,127,127,127,127,99,65,64,76,76,64,127,127,1,103,1,103,103,1,127,0
T 2890 DATA127,127,127,127,127,127,127,127,127,127,99,65,64,76,76,64,127,127,1,103,1,103,103,1,127,0
```

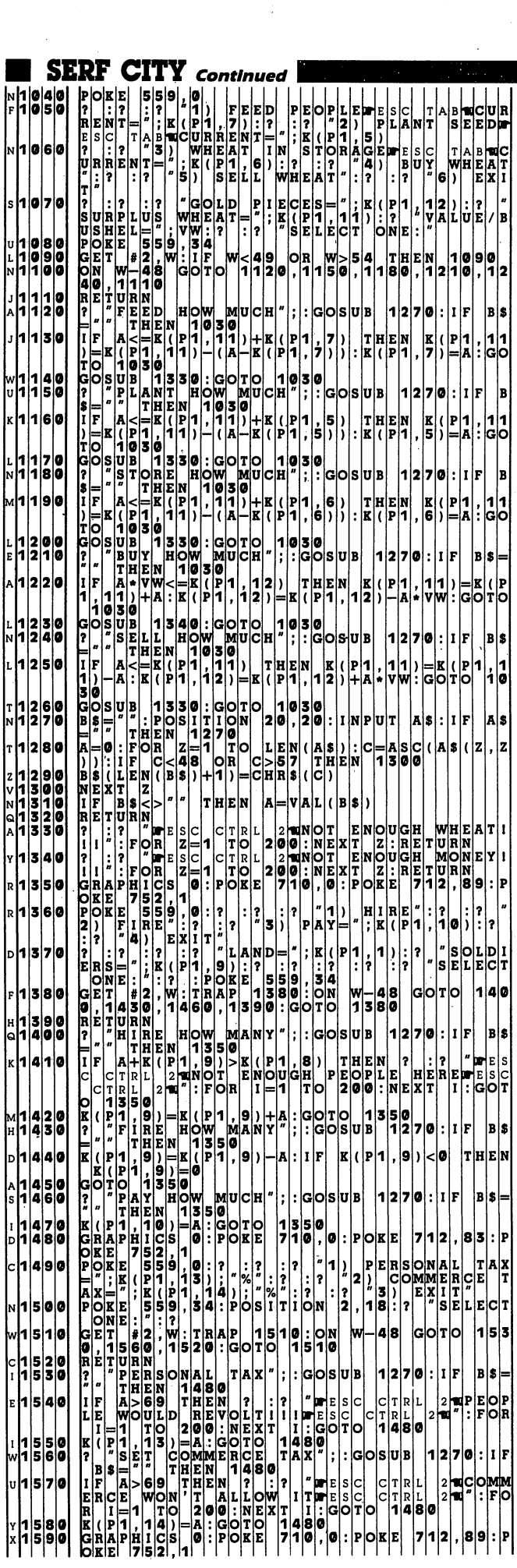
HCM

SERF CITY

ATARI 800/800XL/130XE

```
N 100 REM ***************
N 110 REM * SERF CITY *
T 120 REM ***************
M 130 REM COPYRIGHT 1985
R 140 REM EMERALD VALLEY PUBLISHING CO.
D 150 REM BY WILLIAM K. BALTHROP
L 160 REM AND THE HCM STAFF
K 170 REM HOME COMPUTER MAGAZINE
F 180 REM VERSION 5.6.1
C 190 REM ATARI BASIC FOR THE 800, 800XL, 130XE
F 200 DIM P1$(16),P2$(16),K$(1),K(2,15),A$(41),B$(41)
B 210 RESTORE 240:FOR Z=1 TO 15
C 220 READ A:K(1,Z)=A:READ A:K(2,Z)=A:NEXT Z
S 230 READ P1,P2,VW,YR
E 240 DATA 12,12,2,2,1,1,9,9,0,0,0,0,0,0,90,90,0,0,0,0,500,500,500,500,0,0,0,0,0,0
T 250 DATA 1,2,3,985
A 260 OPEN #2,4,0,"K:"
R 270 GOSUB 2390
G 280 GRAPHICS 2:POSITION 5,3:? #6;"SERF CITY":POKE 752,1
I 290 ?:?"{ESC TAB}(PRESS <RETURN> TO CONTINUE)":GET #2,W
C 300 GRAPHICS 0:POKE 710,0:POKE 712,200
H 310 POSITION 2,5:?"LOAD OLD GAME(Y/N)?";:GET #2,W:IF W=ASC("Y") THEN W=76:GOSUB 2150:GOTO 380
G 320 IF W<>ASC("N") THEN 310
W 330 ?"{ESC CTRL <}":POSITION 2,7:?"ENTER PLAYERS' NAMES:{ESC CTRL =}":?
T 340 POSITION 2,9:?"PLAYER #1:";:INPUT P1$:PL1=LEN(P1$):IF PL1=0 THEN 340
Q 350 IF PL1<16 THEN P1$(PL1+1)=""
J 360 P1$(16)="":POSITION 2,11:?"PLAYER #2:";:INPUT P2$:PL2=LEN(P2$):IF PL2=0 THEN 360
X 370 IF PL2<16 THEN P2$(PL2+1)=""
H 380 P2$(16)="":GRAPHICS 0
Z 390 GOSUB 420:IF DN THEN 1990
M 400 IF P1=2 THEN GOSUB 1590:IF DN THEN 1990
A 410 P1=3-P1:P2=3-P2:GOTO 390
W 420 GOSUB 2280:GET #2,W
M 430 IF W=27 THEN DN=1:RETURN
R 440 IF W=155 THEN RETURN
M 450 IF W=77 OR W=109 THEN GOSUB 500:GOTO 420
C 460 IF W=87 OR W=119 THEN GOSUB 1030:GOTO 420
A 470 IF W=83 OR W=115 THEN GOSUB 1350:GOTO 420
N 480 IF W=84 OR W=116 THEN GOSUB 1480:GOTO 420
C 490 GOTO 420
Z 500 GRAPHICS 0:POKE 703,4:?#6;"{ESC CTRL <}";:?"{ESC CTRL <}";
R 510 IF P1=1 THEN FOR ZZ=1 TO LEN(P1$):S$(802+ZZ,802+ZZ)=CHR$(ASC(P1$(ZZ,ZZ))-32):NEXT ZZ:GOTO 530
M 520 FOR ZZ=1 TO LEN(P2$):S$(802+ZZ,802+ZZ)=CHR$(ASC(P2$(ZZ,ZZ))-32):NEXT ZZ
N 530 POKE 560,0:POKE 561,6:POKE 708,86:POKE 709,198:POKE 710,250:POKE 712,250:POKE 711,52
U 540 A=ADR(S$):POKE 1541,INT(A/256):POKE 1540,A-INT(A/256)*256:POKE 756,CH:L=380
H 550 POKE 764,255:POKE 53279,0:I=ASC(S$(L,L)):C=8
N 560 IF C=8 THEN S$(L,L)="{CTRL G}"
H 570 IF C=16 THEN S$(L,L)=CHR$(I):C=0
Z 580 C=C+1:IF PEEK(764)=14 THEN S$(L,L)=CHR$(I):L=L-40+40*(L<41):GOTO 550
G 590 IF PEEK(764)=15 THEN S$(L,L)=CHR$(I):L=L+40-40*(L>720):GOTO 550
M 600 IF PEEK(764)=6 THEN S$(L,L)=CHR$(I):L=L-1+(L-INT(L/40)*40=1):GOTO 550
R 610 IF PEEK(764)=7 THEN S$(L,L)=CHR$(I):L=L+1-(L-INT(L/40)*40=0):GOTO 550
J 620 IF PEEK(764)=12 THEN S$(L,L)=CHR$(I):POKE 764,255:RETURN
H 630 IF PEEK(764)=0 THEN GOSUB 670:IF P THEN S$(L,L)=CHR$(P1):GOTO 550
R 640 IF PEEK(764)=18 THEN GOSUB 820:IF P THEN S$(L,L)=CHR$(130+P1):GOTO 550
K 650 IF PEEK(764)=37 THEN GOSUB 870:IF P THEN S$(L,L)=CHR$(132+P1):GOTO 550
Y 660 GOTO 560
Q 670 IF K(P1,12)<VW*25 THEN 760
S 680 IF L<41 THEN 700
E 690 P=ASC(S$(L-40)):IF P1-(P/2=INT(P/2))=1 AND P THEN 770
C 700 IF L-INT(L/40)*40=1 THEN 720
C 710 P=ASC(S$(L-1)):IF P1-(P/2=INT(P/2))=1 AND P THEN 770
B 720 IF L>760 THEN 740
P 730 P=ASC(S$(L+40)):IF P1-(P/2=INT(P/2))=1 AND P THEN 770
V 740 IF L-INT(L/40)*40=0 THEN 760
G 750 P=ASC(S$(L+1)):IF P1-(P/2=INT(P/2))=1 AND P THEN 770
L 760 P=0:?"{ESC CTRL 2}";:POKE 764,255:RETURN
F 770 IF I/2=INT(I/2)+(0.5-(P1=1)*0.5) AND I THEN GOSUB 920:IF P<>P1 THEN P=0
D 780 IF I/2=INT(I/2)+(0.5-(P1=2)*0.5) AND I THEN P=0
C 790 IF P THEN K(P1,12)=K(P1,12)-VW*25:K(P1,4)=K(P1,4)+1:K(P1,1)=K(P1,1)+1
U 800 IF P=0 THEN POKE 764,255:?"{ESC CTRL 2}";
J 810 RETURN
A 820 IF P1<>I OR K(P1,12)<500 THEN P=0:?"{ESC CTRL 2}";:POKE 764,255:RETURN
C 830 K(P1,12)=K(P1,12)-500
Z 840 K(P1,2)=K(P1,2)+1
E 850 K(P1,4)=K(P1,4)-1
Y 860 P=1:RETURN
N 870 IF P1<>I OR K(P1,12)<1000 THEN P=0:?"{ESC CTRL 2}";:POKE 764,255:RETURN
R 880 K(P1,12)=K(P1,12)-1000
E 890 K(P1,3)=K(P1,3)+1
W 900 K(P1,4)=K(P1,4)-1
O 910 P=1:RETURN
M 920 P=K(P1,9)*K(P1,10)/K(P1,1)-K(P2,9)*K(P2,10)/K(P2,1)
H 930 FOR Z=6 TO 0 STEP -1:SOUND 0,50,0,Z:FOR A=1 TO 25:POKE 712,PEEK(53770):NEXT A:NEXT Z
T 940 POKE 712,250
F 950 LS=K(P1,9)*0.4*RND(0):K(P1,9)=INT(K(P1,9)-LS):K(P1,8)=INT(K(P1,8)-LS)
F 960 LS=K(P2,9)*0.4*RND(0):K(P2,9)=INT(K(P2,9)-LS):K(P2,8)=INT(K(P2,8)-LS)
U 970 IF P>0 THEN GOTO 1000
Z 980 IF P=0 THEN P=INT(RND(0)*2)+1:IF P=P1 THEN 1000
P 990 P=P2:RETURN
T 1000 P=P1:IF I=131 OR I=132 THEN K(P2,2)=K(P2,2)-1:K(P2,1)=K(P2,1)-1:RETURN
K 1010 IF I=133 OR I=134 THEN K(P2,3)=K(P2,3)-1:K(P2,1)=K(P2,1)-1:RETURN
C 1020 IF I=1 OR I=2 THEN K(P2,4)=K(P2,4)-1:RETURN
Y 1030 GRAPHICS 0:POKE 710,0:POKE 712,67:POKE 752,1
```

Continued



Continued

SERF CITY *Continued*

ATARI 800/800XL/130XE

```
B 2180 ? CHR$(P);
A 2190 IF PP=27 THEN DN=0:? "{ESC}{CTRL <}";:CLOSE #7:RETURN
P 2200 NEXT I:GOTO 2220
M 2210 A$="C:"
A 2220 TRAP 2150:OPEN #7,4+(W=83)*4,0,A$:?
A 2230 ADD=ADR(S$):RWF=7:IF W=83 THEN RWF=11
Q 2240 NUM=800:CHN=7:GOSUB 2610:ADD=ADR(K$):NUM=289:CHN=7:GOSUB 2610
Y 2250 IF W=76 THEN GET #7,VW:GET #7,YR:YR=YR+985:INPUT #7,P1$:INPUT #7,P2$:INPUT #7,PL1:INPUT #7,PL2
O 2260 IF W=83 THEN PUT #7,VW:PUT #7,YR-985:PRINT #7;P1$:PRINT #7;P2$:PRINT #7;PL1:PRINT #7;PL2
U 2270 CLOSE #7:DN=0:RETURN
O 2280 GRAPHICS 0:POKE 710,0:POKE 712,128:POKE 752,1
W 2290 A$=P1$:IF P1=2 THEN A$=P2$
O 2300 ? :? A$:POSITION 30,1:? YR:? :? "TOTAL LAND{2 ESC TAB}";K(P1,1):? "COMMERCE{2 ESC TAB}";K(P1,2):? "MILLS{2 ESC TAB}";K(P1,3)
V 2310 ? "PLANTABLE LAND{ESC TAB}";K(P1,4):? "SEED PLANTED{2 ESC TAB}";K(P1,5):? "WHEAT STORAGE{ESC TAB}";K(P1,6)
W 2320 ? "FOOD{3 ESC TAB}";K(P1,7):? "POPULATION{2 ESC TAB}";K(P1,8):? "SOLDIERS{2 ESC TAB}";K(P1,9):? "SOLDIERS' PAY{ESC TAB}";K(P1,10)
A 2330 ? "WHEAT SURPLUS{ESC TAB}";K(P1,11):? "GOLD PIECES{2 ESC TAB}";K(P1,12):? "PERSONAL TAX{2 ESC TAB}";K(P1,13)
U 2340 ? "COMMERCE TAX{2 ESC TAB}";K(P1,14):? "SCORE{2 ESC TAB}";K(P1,15)
T 2350 IF DN THEN 2380
Q 2360 ? "{37 CTRL R}"
C 2370 ? "  M)MAP   W)WHEAT   S)SOLDIERS   T)TAXES":? :? "   (PRESS M W S T <ESC> OR <RETURN>)"
H 2380 RETURN
W 2390 DATA 170,170,170,170,170,170,170,170
E 2400 DATA 85,85,85,85,85,85,85,85
B 2410 DATA 190,255,255,255,239,239,239,255
R 2420 DATA 125,255,255,255,223,223,223,255
N 2430 DATA 186,254,186,254,186,186,186,186
O 2440 DATA 117,253,117,253,117,117,117,117
P 2450 DATA 0,0,60,60,60,60,0,0
A 2460 DATA 170,85,170,85,170,85,170,85
U 2470 DIM ML$(40):ML$="h▨{CTRL E}L{CTRL E}N(—t▨{CTRL B}▨{CTRL E}M){CTRL .}{CTRL E}O1N{CTRL Q}LHPy{M{Oh{CTRL .}▨{CTRL D}▨Pp▨{CTRL .}"
Q 2480 DIM S$(960):S$="{CTRL ,}":S$(960)="{CTRL ,}":S$(2)=S$:S$(761)="{40 CTRL H}"
S 2490 S$(821)="{2 CTRL ,}—/6%{CTRL ,}7)4({2 CTRL ,}{SHIFT +}{CTRL ,}↑{CTRL ,}—{CTRL ,}]{2 CTRL ,},{CTRL ,}{CTRL I}{CTRL ,},!.$ {2 CTRL ,}#{CTRL I}{CTRL ,},#/——%2#%{2 CTRL ,}—{CTRL I}{CTRL ,},—),,{11 CTRL ,}▨2%452.▨{CTRL ,}4/{CTRL ,},%8)4"
Z 2500 S$(960)="{CTRL ,}"
Y 2510 POKE 106,PEEK(106)-8:GRAPHICS 0:? "INITIALIZING...";:POKE 756,PEEK(106)+4:I=USR(ADR(ML$)):CH=PEEK(756)*256
E 2520 RESTORE 2580:FOR I=8 TO 11:READ A$:S$(I*40-39,I*40)=A$:NEXT I
H 2530 RESTORE 2390:FOR I=CH+8 TO CH+71:READ A:POKE I,A:NEXT I:CH=CH/256
D 2540 DL=PEEK(560)+PEEK(561)*256
G 2550 POKE 1536,112:POKE 1537,112:POKE 1538,112:POKE 1539,68:POKE 1540,PEEK(DL+4):POKE 1541,PEEK(DL+5)
G 2560 FOR I=1542 TO 1560:POKE I,4:NEXT I:FOR I=1561 TO 1564:POKE I,2:NEXT I:POKE 1565,65:POKE 1566,0:POKE 1567,6
E 2570 RETURN
V 2580 DATA {CTRL ,},{CTRL A}▨{CTRL E}▨{CTRL A}{32 CTRL ,},{CTRL B}▨{CTRL F}▨{CTRL B}{CTRL ,},{CTRL ,},{CTRL A}▨{CTRL C}▨{CTRL A}{32 CTRL ,},{CTRL B}▨{CTRL D}▨{CTRL B}{CTRL ,}
V 2590 DATA {CTRL ,},{CTRL A}▨{CTRL C}▨{CTRL A}{32 CTRL ,},{CTRL B}▨{CTRL D}▨{CTRL B}{CTRL ,}
R 2600 DATA {CTRL ,},{3 CTRL A}{32 CTRL ,},{3 CTRL B}{CTRL ,}
P 2610 CHN=16*CHN
P 2620 IOCB=832+CHN:POKE IOCB+2,RWF
Y 2630 ADH=INT(ADD/256):ADL=ADD-ADH*256
L 2640 POKE IOCB+4,ADL:POKE IOCB+5,ADH
G 2650 NMH=INT(NUM/256):NML=NUM-NMH*256
D 2660 POKE IOCB+8,NML:POKE IOCB+9,NMH
D 2670 A=USR(ADR("hhh▨*▨LV▨d▨"),CHN)
C 2680 RETURN
```

HCM

SERF CITY

COMMODORE 64

```
N 100 REM ***************
N 110 REM *  SERF CITY  *
T 120 REM ***************
M 130 REM COPYRIGHT 1985
R 140 REM EMERALD VALLEY PUBLISHING CO.
D 150 REM BY WILLIAM K. BALTHROP
L 160 REM AND THE HCM STAFF
K 170 REM HOME COMPUTER MAGAZINE
F 180 REM VERSION 5.6.1
B 190 REM COMMODORE 64 BASIC
F 200 POKE56,72:CLR:C=RND(-TI):POKE53280,14:POKE53281,0
L 210 POKE53272,21:POKE657,128:POKE53272,PEEK(53272)OR21
J 220 PRINT"{SHIFT CLR}"TAB(16)"{CTRL WHT}{11 CRSR DOWN}SERF CITY{11 CRSR DOWN}"
K 230 PRINTTAB(9)"PRESS {CTRL RVS ON}RETURN{CTRL RVS OFF} TO CONTINUE";:GOSUB2890
Z 240 PRINT"{SHIFT CLR}{6 CRSR DOWN}      PLEASE WAIT WHILE I INITIALIZE.":GOSUB280
X 250 PRINT"{CRSR DOWN} THE SCREEN WILL BLANK FOR A WHILE, SO":PRINTTAB(14)"{CTRL RVS ON}DON'T PANIC!
B 260 FORZ=1TO3200:NEXT:PRINTTAB(15)"{CRSR DOWN}SEE YOU...":FORZ=1TO1900:NEXT
M 270 POKE53265,PEEK(53265)AND239:GOSUB2670:GOTO2940
C 280 DIMP$(2),K(2,15):FORZ=1TO14:READK(1,Z),K(2,Z):NEXT
C 290 READP1,P2,VW,YR,I1$,I2$,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9
E 300 DATA12,12,2,2,1,1,9,9,0,0,0,0,0,0,90,90,0,0,0,0,500,500,500,500,0,0,0,0,1
Q 310 DATA2,3,985,MMWWSSTT,ESDXLCM,0,1,2,3,4,5,6,7,8,9:RETURN
G 320 PRINT:PRINT"{SHIFT CLR}{7 CRSR DOWN}":PRINT"ENTER PLAYER NAMES:{3 CRSR DOWN}"
H 330 INPUT"PLAYER #1:  ";P$(A1)
H 340 P$(A1)=MID$(P$(A1),A1,12):IFP$(A1)=""THENP$(A1)="PLAYER ONE"
N 350 INPUT"PLAYER #2:  ";P$(A2)
N 360 P$(A2)=MID$(P$(A2),A1,12):IFP$(A2)=""THENP$(A2)="PLAYER TWO"
Y 370 PRINT"{SHIFT CLR}":GOSUB3230
O 380 GOSUB430:IFDNTHEN2370
M 390 IFP1=A2THEN410
G 400 P1=A2:P2=A1:GOTO380
T 410 GOSUB1830:IFDNTHEN2370
U 420 P2=A2:P1=A1:GOTO380
C 430 GOSUB2500
X 440 GOSUB2890:IFW=133THENGOSUB3050:POKE198,0:GOTO430
V 450 IFW<>136THEN470
L 460 DN=-1:RETURN
L 470 IFW<>13THEN490
N 480 PRINT"{SHIFT CLR}":C1=3:C2=4:C3=16:GOSUB2920:RETURN
T 490 A=A0:FORZ=A1TOA8:IFMID$(I1$,Z,A1)=CHR$(W)THENA=Z
D 500 NEXT:IFA=A0THEN440
Y 510 ONINT(A*.5+.6)GOSUB520,1140,1530,1690:POKE650,0:GOTO430
L 520 POKE646,0:POKE650,128:GOSUB2660:POKE53281,5:X=20:Y=11:P=1
X 530 C=PEEK(V+X+Y*40)
X 540 GETW$:IFP=A0THEN560
H 550 POKE(V+X+Y*40),C:GOTO570
E 560 POKE(V+X+Y*40),222
M 570 P=ABS(P-A1):IFW$=""THEN540
A 580 POKE(V+X+Y*40),C:P=A0:IFASC(W$)<>13THEN600
V 590 PRINT"{SHIFT CLR}":C1=3:C2=4:C3=16:GOSUB2920:POKE53281,0:POKE646,1:RETURN
Q 600 A=A0:FORZ=A1TOA7:IFMID$(I2$,Z,A1)=W$THENA=Z
I 610 NEXT:IFA=A0THEN540
X 620 ONAGOSUB630,660,690,720,750,1050,1060:GOTO530
K 630 IFY>A0THEN650
Y 640 RETURN
```

Continued

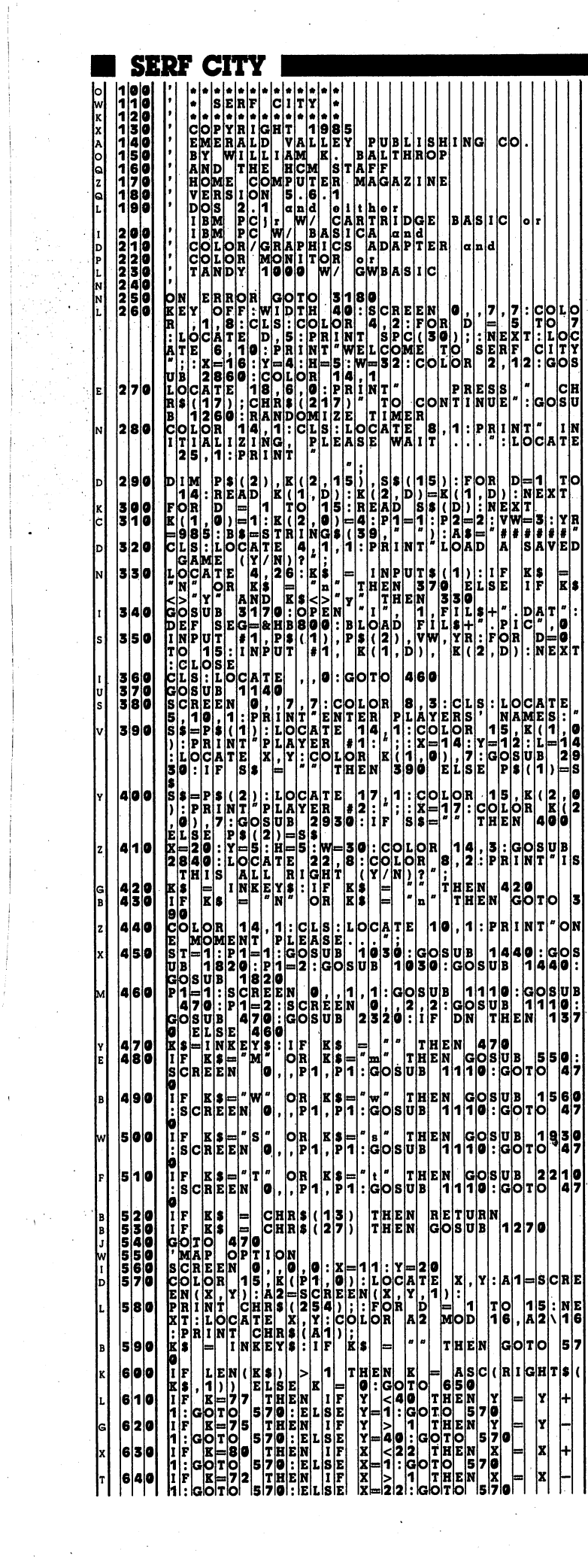
```
H  650 Y=Y-A1:RETURN
B  660 IFX>A0THEN680
R  670 RETURN
U  680 X=X-A1:RETURN
H  690 IFX<39THEN710
N  700 RETURN
I  710 X=X+A1:RETURN
R  720 IFY<20THEN740
E  730 RETURN
L  740 Y=Y+A1:RETURN
C  750 IFK(P1,12)<VW*25THEN880
L  760 C1=0:C2=0:C3=0:C4=0:IFY>0THENC1=PEEK(V+X+(Y-1)*40)
X  770 IFX<39THENC2=PEEK(V+X+1+Y*40)
L  780 IFY<19THENC3=PEEK(V+X+(Y+1)*40)
D  790 IFX>0THENC4=PEEK(V+X-1+Y*40)
H  800 IF(P1<>A1)OR(C<>32ANDC<>160)THEN840
M  810 IF(C1=96ORC2=96ORC3=96ORC4=96)THEN890
P  820 IF(C1=93ORC2=93ORC3=93ORC4=93)THEN890
M  830 IF(C1=92ORC2=92ORC3=92ORC4=92)THEN890
Q  840 IF(P1<>A2)OR(C<>32ANDC<>96)THEN880
A  850 IF(C1=160ORC2=160ORC3=160ORC4=160)THEN890
Z  860 IF(C1=157ORC2=157ORC3=157ORC4=157)THEN890
C  870 IF(C1=156ORC2=156ORC3=156ORC4=156)THEN890
N  880 POKE54276,21:GOSUB2910:POKE54276,0:RETURN
T  890 IFC=32THEN920
Y  900 GOSUB970:IFP=P1THEN920
Q  910 RETURN
L  920 IFP1=A2THEN940
B  930 CH=96:GOSUB1130:GOTO950
N  940 CH=160:GOSUB1130
Y  950 K(P1,12)=K(P1,12)-VW*25:K(P1,A1)=K(P1,A1)+A1
L  960 K(P1,A4)=K(P1,A4)+A1:RETURN
F  970 P=K(P1,A9)*K(P1,10)/K(P1,A1)-K(P2,A9)*K(P2,10)/K(P2,A1)
A  980 POKE54276,21:GOSUB2910:POKE54276,32:GOSUB2910:POKE54276,0
N  990 K(P1,A9)=K(P1,A9)-INT(K(P1,A9)*.5*RND(0))
N 1000 K(P2,A9)=K(P2,A9)-INT(K(P2,A9)*.4*RND(0)):IFP>A0THEN1030
N 1010 IFP=A0THEN1040
K 1020 P=P2:RETURN
K 1030 P=P1:RETURN
L 1040 P=INT(RND(0)*A2)+A1:RETURN
A 1050 C1=500:C2=93:C3=157:C4=2:GOSUB1070:RETURN
W 1060 C1=1000:C2=92:C3=156:C4=3:GOSUB1070:RETURN
P 1070 IF(P1=A1ANDC<>96)OR(P1=A2ANDC<>160)ORK(P1,12)<C1ORC=32THEN880
C 1080 IFP1=A2THEN1100
Q 1090 CH=C2:GOSUB1130:GOTO1110
N 1100 CH=C3:GOSUB1130
T 1110 K(P1,12)=K(P1,12)-C1:K(P1,C4)=K(P1,C4)+A1
G 1120 K(P1,A4)=K(P1,A4)-A1:RETURN
Z 1130 POKE(V+X+Y*40),CH:RETURN
L 1140 PRINT"{SHIFT CLR}{3 CRSRDOWN}":PRINT"1) FEED PEOPLE";TAB(25);"CURRENT=";K(P1,A7)
K 1150 PRINT"2) PLANT SEED";TAB(25);"CURRENT=";K(P1,A5)
W 1160 PRINT"3) WHEAT IN STORAGE";TAB(25);"CURRENT=";K(P1,A6)
I 1170 PRINT"4) BUY WHEAT":PRINT"5) SELL WHEAT":PRINT"6) EXIT
V 1180 PRINT:PRINT"GOLD PIECES=     ";K(P1,12)
X 1190 PRINT"SURPLUS WHEAT=  ";K(P1,11):PRINT"VALUE/BUSHEL=  ";VW
X 1200 PRINT:PRINT"SELECT ONE
K 1210 GOSUB2890:IFW<49ORW>54THEN1210
U 1220 ONW-48GOTO1230,1270,1310,1350,1390:RETURN
V 1230 GOSUB1430:IFB$=""THEN1140
K 1240 IFA<=(K(P1,11)+K(P1,A7))THEN1260
W 1250 GOSUB1510:GOTO1140
H 1260 K(P1,11)=K(P1,11)-(A-K(P1,A7)):K(P1,A7)=A:GOTO1140
H 1270 GOSUB1430:IFB$=""THEN1140
J 1280 IFA<=(K(P1,11)+K(P1,A5))THEN1300
Q 1290 GOSUB1510:GOTO1140
K 1300 K(P1,11)=K(P1,11)-(A-K(P1,A5)):K(P1,A5)=A:GOTO1140
I 1310 GOSUB1430:IFB$=""THEN1140
B 1320 IFA<=(K(P1,11)+K(P1,A6))THEN1340
Z 1330 GOSUB1510:GOTO1140
I 1340 K(P1,11)=K(P1,11)-(A-K(P1,A6)):K(P1,A6)=A:GOTO1140
K 1350 GOSUB1430:IFB$=""THEN1140
H 1360 IFA*VW<=K(P1,12)THEN1380
W 1370 GOSUB1520:GOTO1140
R 1380 K(P1,11)=K(P1,11)+A:K(P1,12)=K(P1,12)-(A*VW):GOTO1140
G 1390 GOSUB1430:IFB$=""THEN1140
Z 1400 IFA<=K(P1,11)THEN1420
I 1410 GOSUB1510:GOTO1140
E 1420 K(P1,11)=K(P1,11)-A:K(P1,12)=K(P1,12)+(A*VW):GOTO1140
R 1430 A$="":INPUT"HOW MUCH  ";A$
V 1440 B$="":A=A0:IFA$=""THENRETURN
L 1450 FORZ=A1TOLEN(A$):C=ASC(MID$(A$,Z,A1)):IFC<48ORC>57THEN1470
Q 1460 B$=B$+CHR$(C):IFZ>A6THEN1480
Z 1470 NEXT
V 1480 IFB$<>""THEN1500
O 1490 RETURN
O 1500 A=VAL(B$):RETURN
P 1510 PRINT"{3 CRSRDOWN}NOT ENOUGH WHEAT!!!{3 CRSRDOWN}":GOSUB2880:RETURN
T 1520 PRINT"{3 CRSRDOWN}NOT ENOUGH MONEY!!!{3 CRSRDOWN}":GOSUB2880:RETURN
P 1530 PRINT"{SHIFT CLR}":PRINT"{5 CRSRDOWN}1) HIRE":PRINT"2) FIRE
S 1540 PRINT"3) PAY:        ";K(P1,10):PRINT"4) EXIT
L 1550 PRINT:PRINT"LAND=        ";K(P1,A1):PRINT:PRINT"SOLDIERS=";K(P1,A9)
N 1560 PRINT:PRINT:PRINT"SELECT ONE
A 1570 GOSUB2890:IFW<49ORW>52THEN1570
T 1580 ONW-48GOTO1600,1640,1670,1590
T 1590 RETURN
I 1600 INPUT"HIRE:";A$:GOSUB1440:IFB$=""THEN1530
T 1610 IF(A+K(P1,A9))<=K(P1,A8)THEN1630
Y 1620 PRINT"NOT ENOUGH PEOPLE HERE{2 CRSRDOWN}":GOSUB2880:GOTO1530
T 1630 K(P1,A9)=K(P1,A9)+A:GOTO1530
A 1640 INPUT"FIRE:";A$:GOSUB1440:IFB$=""THEN1530
N 1650 K(P1,A9)=K(P1,A9)-A:IFK(P1,A9)=A0THEN1530
O 1660 K(P1,A9)=A0:GOTO1530
T 1670 INPUT"PAY:";A$:GOSUB1440:IFB$=""THEN1530
N 1680 K(P1,10)=A:GOTO1530
T 1690 PRINT"{SHIFT CLR}":PRINT"{6 CRSRDOWN}1) PERSONAL TAX=  ";K(P1,13);"%
N 1700 PRINT"{CRSRDOWN}2) COMMERCE TAX=  ";K(P1,14);"%":PRINT"{CRSRDOWN}3) EXIT
F 1710 PRINT"{CRSRDOWN}SELECT ONE:
L 1720 GOSUB2890:IFW<49ORW>51THEN1720
N 1730 ONW-48GOTO1750,1790,1740
O 1740 RETURN
L 1750 INPUT"PERSONAL:";A$:GOSUB1440:IFB$=""THEN1690
Y 1760 IFA<70THEN1780
Q 1770 PRINT"PEOPLE WOULD REVOLT!!!":GOSUB2880:GOTO1690
E 1780 K(P1,13)=A:GOTO1690
K 1790 INPUT"COMMERCE:";A$:GOSUB1440:IFB$=""THEN1690
I 1800 IFA<70THEN1820
S 1810 PRINT"COMMERCE WON'T LET YOU":GOSUB2880:GOTO1690
W 1820 K(P1,14)=A:GOTO1690
S 1830 FORZ=A1TOA2:POKE53284,PEEK(53281+Z):POKE53280,PEEK(53284):PRINT"{SHIFT CLR}
K 1840 PRINTP$(Z);"S' HARVEST{CRSRDOWN}":IFK(Z,A5)<=K(Z,A4)*10THEN1860
J 1850 K(Z,A5)=K(Z,A4)*10
A 1860 E=K(Z,A5):IFE<=K(Z,A8)-K(Z,A9)THEN1880
Q 1870 E=K(Z,A8)-K(Z,A9):K(Z,A5)=E
T 1880 F=INT(K(Z,A3)*((70-K(Z,14))*.2)+.01)
M 1890 IFF<=K(Z,A8)-E-K(Z,A9)THEN1910
T 1900 F=K(Z,A8)-E-K(Z,A9)
R 1910 G=INT(K(Z,A2)*((120-K(Z,14))*.2))
P 1920 IFG<=K(Z,A8)-E-F-K(Z,A9)THEN1940
K 1930 G=K(Z,A8)-E-F-K(Z,A9)
S 1940 R=RND(0):PRINT"EMPLOYMENT{CRSRDOWN}":PRINT"WHEAT        ";E:PRINT"MILL      ";F
L 1950 PRINT"COMMERCE    ";G;"{CRSRDOWN}":P=INT(R*17+A3)*K(Z,A5)
E 1960 PRINT"HARVEST=     ";P;"{CRSRDOWN}":C1=INT(K(Z,11)*RND(0))
X 1970 K(Z,11)=K(Z,11)+P-C1:C2=INT(RND(0)*K(Z,A6)*.1)
C 1980 K(Z,A6)=K(Z,A6)-C2:C1=C1+C2:PRINT"RATS ATE  ";C1;" BUSHELS{CRSRDOWN}
U 1990 P=K(Z,A7)/(K(Z,A8)*A5)*((F+A1)/(K(Z,A8)*.1))*ABS((110-K(Z,13))*.01)
N 2000 C1=INT((G+E+F)*K(Z,13)*P*.15):C2=INT((F+G)*K(Z,14)*P*.25)
V 2010 PRINT"PERSONAL TAX=  ";C1:PRINT"COMMERCE TAX=  ";C2;"{CRSRDOWN}
W 2020 K(Z,12)=K(Z,12)+C1+C2:C1=A0
R 2030 IF(K(Z,A9)=A0)AND(K(Z,10)=A0)THEN2110
Y 2040 IFK(Z,10)>A2THEN2080
U 2050 IFK(Z,10)>A0THEN2070
I 2060 C2=K(Z,A9):GOTO2100
```

Continued

SERF CITY *Continued*


```
Y 2070  C1=INT(K(Z,A9))/(K(Z,10)+A1)):K(Z,A9)=K(Z,A9)-C1
O 2080  C3=K(Z,A9)*K(Z,10):IFC3<=K(Z,12)THEN2110
U 2090  C2=INT(K(Z,12)/K(Z,10)):C2=K(Z,A9)-C2
H 2100  K(Z,A9)=K(Z,A9)-C2:C1=C1+C2
V 2110  PRINTC1;" SOLDIERS DESERTED":K(Z,A8)=K(Z,A8)-C1
C 2120  PRINTTAB(9);"SOLDIERS PAY= ";K(Z,A9)*K(Z,10);"■CRSRDOWN■"
G 2130  K(Z,12)=K(Z,12)-(K(Z,A9)*K(Z,10))
L 2140  K(Z,INT(K(Z,A8)*.05*PRND(0))):IFC1<(K(Z,INT(A8)*.5)THEN2160
Z 2150  K(Z,INT(K(Z,A8)*.5)
X 2160  K(Z,A8)=K(Z,A8)+C1:C2=INT(K(Z,A8)*.2/(P*RND(0)+1))
H 2170  IFC2<=K(Z,A8)THEN2190
C 2180  C2=K(Z,A8)
C 2190  K(Z,A8)=K(Z,A8)-C2:IFP<A1THEN2230
N 2200  C3=INT(RND(0)*(K(P1,A7)*.2-K(P1,A8)))
B 2210  C3=A0:PRINTC3;" PEOPLE MOVED HERE":GOTO222
A 2220  C3=INT(K(Z,A8)*P*RND(0))-K(Z,A8))
Z 2230  C3=INT((K(Z,A8)*P*RND(0))-K(Z,A8))
A 2240  PRINTC3*.1;" PEOPLE MOVED AWAY"
X 2250  PRINTC1;" PEOPLE BORN":PRINTC2;" PEOPLE■2 CRSRDOWN■"
Q 2260  K(Z,A8)=K(Z,A8)+C3:IFK(Z,A9)<=K(Z,A8)THEN2280
F 2270  K(Z,A8)=K(Z,A8)
Y 2280  PRINT"PRESS RETURN TO CONTINUE";:IF
Z 2290  >1THENPRINT" OR F1 TO SAVE";
F 2290  GOSUB2890:IFW<>13ANDW<>133THEN2290
R 2300  K(Z,A5)=A0:K(Z,A7)=A0
U 2310  K(Z,11))*.2+K(Z,A8))+K(Z,12)*.3)
S 2320  NEXT:A1=YR+YR+A1
D 2330  ,11)<A1,A8)>A0ANDK(A2,A8)>A0ANDK(A1,15)<100000ANDK(A2,15)<100000THEN2350
U 2340  DN=-1
M 2350  VW=INT(RND(0)*A6)+A1:IFW=133THENGOSUB3050
N 2360  RETURN
O 2370  PRINT"■SHIFT CLR■":PRINTTAB(8);"III WE HAVE A WINNER!!!!■4 CRSRDOWN■PRESS RETURN TO FIND OUT WHO■7 CRSRDOWN■"
T 2380  GOSUB2890:IFW<>13THEN2390
U 2390  FORP1=A1TOA2:GOSUB2500:PRINT"PRESS RETURN TO CONTINUE";
N 2400  NEXT:CLOSE1:CLOSE15:GOSUB3230:GOTO37
U 2410  GOSUB2890:IFW<>13THEN2410
I 2420  NEXT:PRINT"■SHIFT CLR■":PRINT"■6 CRSRDOWN■14 CRSRRIGHT■THE WINNER IS
J 2430  IFK(A1,15)>K(A2,15)THEN2450
F 2440  Z=A2:GOTO2460
I 2450  Z=A1
Q 2460  POKE53284,PEEK(53281+Z):POKE53280,PEEK(53284):PRINTTAB(17);"■6 CRSRDOWN■";
U 2470  PRINT"■8 CRSRDOWN■PRESS RETURN";
S 2480  GOSUB2890:IFW<>13THEN2480
M 2490  GOTO3180
Y 2500  POKE53284,PEEK(53281+P1):POKE53280,PEEK(53284):PRINT"■SHIFT CLR■
Z 2510  PRINT"■4 CRSRDOWN■1 ";TAB(24);YR;"■4 CRSRDOWN■PRINT"TOTAL LAND";TAB(16);K(P1,A1)
Y 2520  PRINT"COMMERCE";TAB(16);K(P1,A2):PRINT"MILLS";TAB(16);K(P1,A3)
S 2530  PRINT"PLANTABLE LAND";TAB(16);K(P1,A4)
V 2540  PRINT"SEED PLANTED";TAB(16);K(P1,A5)
D 2550  PRINT"WHEAT STORAGE";TAB(16);K(P1,A6):PRINT"FOOD";TAB(16);K(P1,A7)
A 2560  PRINT"POPULATION";TAB(16);K(P1,A8):PRINT"SOLDIERS";TAB(16);K(P1,A9)
T 2570  PRINT"SOLDIERS PAY";TAB(16);K(P1,10)
J 2580  PRINT"WHEAT SURPLUS";TAB(16);K(P1,11)
B 2590  PRINT"GOLD PIECES";TAB(16);K(P1,12)
Q 2600  PRINT"PERSONAL TAX";TAB(16);K(P1,13)
R 2610  PRINT"COMMERCE TAX";TAB(16);K(P1,14)
B 2620  PRINT"SCORE";TAB(16);K(P1,15);"■2 CRSRDOWN■";:IFDNTHEN2650
T 2630  PRINT"MAP WHEAT SOLD TAX
V 2640  PRINT"PRESS M W S T F1 F7 OR RETURN";:RETURN
V 2650  RETURN
J 2660  PRINT"■SHIFT CLR■":C1=2:C2=2:C3=0:GOSUB2920:RETURN
B 2670  V=16384:GOSUB2790:FORZ=0TO839:POKEV+Z,32:NEXT
O 2680  FORZ=0TO39:POKE840+V+Z,223:NEXT:FORZ=0TO119:POKE880+V+Z,224:NEXT
M 2690  S$="MOVE WITH E S D X
O 2700  FORZ=1TO17:A=ASC(MID$(S$,Z,1))+128:IFA=160THENNEXT
R 2710  POKE V+929+Z,A:NEXT
J 2720  S$=" L AND C OMERCE M ILL RETURN TO EXIT
P 2730  FORZ=1TO39:A=ASC(MID$(S$,Z,1))+128:IFA=160THENNEXT
D 2740  POKEV+959+Z,A:NEXT:FORZ=0TO3:POKEV+40*Z),96:POKEV+40+(40*Z),96
N 2750  POKEV+43+(40*Z),160:POKEV+436+(40*Z),160:NEXT
L 2760  POKEV+404,92:POKEV+435,156:POKEV+443,93:POKEV+475,157
Y 2770  POKEV+484,93:POKEV+515,157:POKEV+523,96:POKEV+555,160
S 2780  C1=3:C2=4:C3=16:GOSUB2920:RETURN
T 2790  PEEK(56334)AND254:POKE1,PEEK(1)AND251
S 2800  FORZ=0TO511:POKEZ+18432,PEEK(Z+53248):NEXT
I 2810  POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1:FORZ=224TO255:POKEZ,PEEK(Z+53248):NEXT
L 2820  POKEZ+18432,A:NEXT:POKE53282,7:POKE53283,3:POKE53284,14
H 2830  POKE53272,(PEEK(53272)AND240)+2:POKE53265,PEEK(53265)OR64
J 2840  S=PEEK(53272):FORP=0TO+24:POKEZ,A0:NEXT
K 2850  POKES+1,12:FORP=0TOS+5,9:POKES+15,247:POKES+24,15:RETURN
B 2860  DATA0,0,0,255,48,48,8,28,62,127,54,54,54,0
P 2870  DATA0,0,60,60,60,60,0,0,0,255,0,255,0,255
V 2880  FORP=A1TO500:NEXT:RETURN
D 2890  W$=INKEY$:IFW$=""THEN2890
I 2900  W=ASC(W$):RETURN
G 2910  FORP=A1TO15:NEXT:RETURN
R 2920  POKEE56578,PEEK(56578)OR3:POKEE56576,(PEEKE56576)AND252)ORC1
H 2930  POKE53272,(PEEK(53272)AND240)ORC2:POKE53272,(PEEKE53272)AND15)ORC3:RETURN
E 2940  POKE53265,PEEK(53265)OR16:PRINT"■SHIFT CLR■":INPUT"LOAD OLD GAME (Y/N)";A$
X 2950  IFA$<>"Y"ANDA$<>"N"THEN2940
A 2960  IFA$="N"THEN3200
V 2970  INPUT"DISK ";A$:IFA$<>"T"ANDA$<>"D"THEN2940
L 2980  PRINT:INPUT"FILE NAME:";F$:IFA$="D"THENOPEN1,8,2,F$+",S,R":GOTO3010
J 2990  OPEN1,0,0,F$+".SR":GOTO3010
G 3000  OPEN1,8,2,"0:"+F$+".SR"+",S,R":GOSUB3230:GOTO3710
Z 3010  INPUT#1,VW$:VW=VAL(VW$):INPUT#1,YR$:YR=VAL(YR$):INPUT#1,DN$:DN=VAL(DN$)
X 3020  INPUT#1,P$(1)
G 3030  INPUT#1,P$(2):FORZ=1TO2:FORA=1TO15:INPUT#1,K$:K(Z,A)=VAL(K$):NEXT:NEXT
J 3040  FORZ=0TO999:INPUT#1,A:POKE(V+Z),A:NEXT:CLOSE1:CLOSE15:GOSUB3230:GOTO3710
O 3050  PRINT"■SHIFT CLR■":INPUT"SAVE GAME (Y/N)";A$:IFA$<>"Y"ANDA$<>"N"THENRETURN
H 3060  IFA$="N"THENRETURN
A 3070  INPUT"DISK ";A$:IFA$<>"T"ANDA$<>"D"THEN3050
O 3080  PRINT:INPUT"FILE NAME:";F$:IFA$="D"THENOPEN1,8,15,"I":GOTO3110
P 3090  OPEN1,1,1,F$+".SR":GOTO3110
F 3100  OPEN1,8,2,"@:"+F$+".SR"+",S,W":GOSUB3150:CLOSE1:CLOSE15:GOTO3050
B 3110  VW$=STR$(VW):PRINT#1,VW$:YR$=STR$(YR):DN$=STR$(DN):PRINT#1,DN$
N 3120  PRINT#1,P$(1):PRINT#1,P$(2):FORZ=1TO2:FORA=1TO15:K$=STR$(K(Z,A)):PRINT#1,K$:NEXT:NEXT
D 3140  FORZ=0TO999:A=PEEK(V+Z):PRINT#1,A:NEXT:CLOSE1:CLOSE15:RETURN
Y 3150  OPEN15,8,15:INPUT#15,A,A$:IFA=0THENRETURN
G 3160  PRINT"■CRSRDOWN■***"A$"***":FORZ=1TO31:POKE54296,15*(ZAND1):NEXT
M 3170  FORZ=1TO3000:NEXT:CLOSE1:CLOSE15:RETURN
T 3180  POKE53265,PEEK(53265)AND191
Q 3190  POKE198,0:INPUT"■SHIFT CLR■■8 CRSRDOWN■PLAY AGAIN (Y/N)";A$
M 3200  IFA$<>"Y"ANDA$<>"N"THEN3190
K 3210  IFA$="Y"THENRUN
S 3220  END
Z 3230  S$=P$(A1):FORZ=1TOLEN(S$):A=ASC(MID$(S$,Z,1))+128:IFA<161THENNEXT
O 3240  POKEV+879+Z,A:NEXT
W 3250  S$=P$(A2):L=LEN(S$):FORZ=1TOL:A=ASC(MID$(S$,L+1-Z,1))+128:IFA<161THENNEXT
U 3260  POKEV+899+21-Z,A:NEXT:RETURN
```

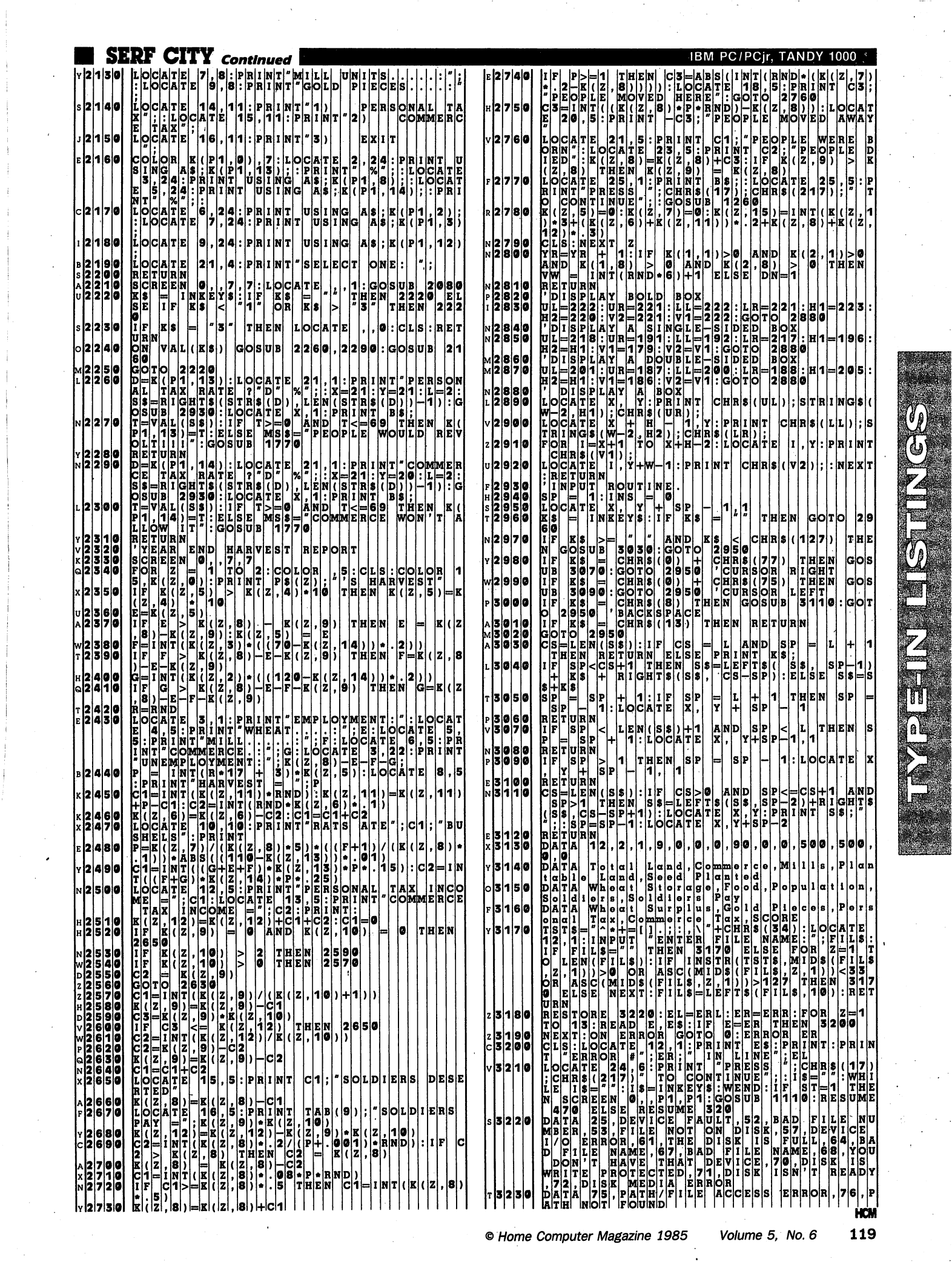
HCM



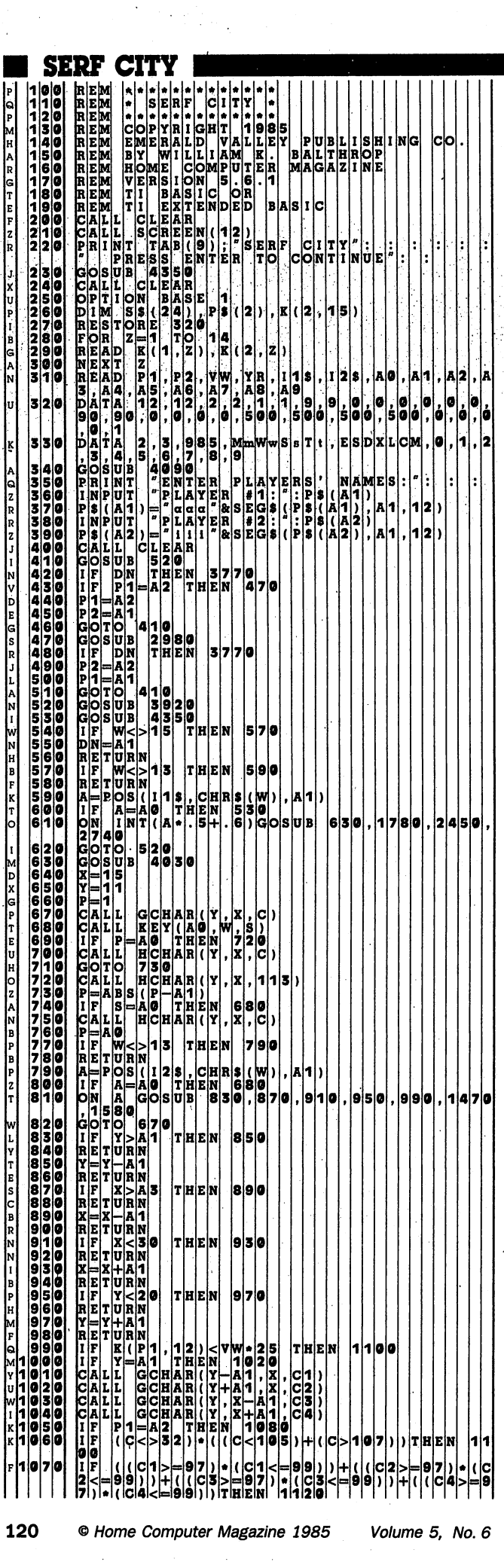
Continued

```
P 12300 LOCATE 25,1:PRINT"PRESS  ANY  KEY  TO  EXIT";:A$=INPUT$(1):CLS:SCREEN 0,0:WIDTH 80:KEY ON:CHAIN"MENU"
N 12400 CHR$(217),,7:RETURN
S 12500 RETURN INKEY$:IF A$="" THEN 12500 ELSE RETURN
L 12600 KEY$=INKEY$:IF KEY$="" THEN RETURN ELSE IF LEN(KEY$)=1 THEN RETURN ELSE KEY=ASC(RIGHT$(KEY$,1)):RETURN
Y 12700 GAME=2:GOTO 13700
Z 12800 GAME=1:GOTO 13700
M 12900 IF CW=0 THEN CW=6:CLS:LOCATE 14,2:PRINT"Y/N";:GOSUB 12500:IF KEY$="Y" THEN CW=1:RETURN ELSE IF KEY$="N" THEN RETURN ELSE 12900
D 13000 CLS:LOCATE 8,7:PRINT"SAVE  GAME  (Y/N)";:GOSUB 12500:IF KEY$="Y" THEN 13100 ELSE IF KEY$="N" THEN RETURN ELSE 13000
C 13100 OPEN"O",#1,"SERF.DAT":FOR I=1 TO 4:WRITE#1,X(I),Y(I),D(I),K(I):NEXT:WRITE#1,CW,YR,MO,DA,HR,MN,SC:CLOSE:RETURN
M 13200 OPEN"I",#1,"SERF.DAT":FOR I=1 TO 4:INPUT#1,X(I),Y(I),D(I),K(I):NEXT:INPUT#1,CW,YR,MO,DA,HR,MN,SC:CLOSE:RETURN
W 13300 PRINT #1,"";:RETURN
X 13400 SAVE FILE$+".PIC",0,&H8000,&H4000:RETURN
N 13500 PRINT "EXIT  GAME  (Y/N)";:GOSUB 12500:IF KEY$="Y" THEN 13600 ELSE IF KEY$="N" THEN RETURN ELSE 13500
J 13600 KEY$="":INPUT"KEY";KEY$:IF KEY$="" THEN 13600 ELSE IF LEN(KEY$)=1 THEN RETURN ELSE KEY=ASC(RIGHT$(KEY$,1)):RETURN
Q 13700 SCREEN 1,0:DEF SEG=&HB800:BLOAD"SERF.PIC",0:DEF SEG:RETURN
O 13800 CLS:LOCATE 12,17:PRINT"WORKING";:RETURN
P 13900 FOR K=1 TO 2:SCREEN 0,0,K,7:CLS:PRINT"SCREEN 0,0,"K",7":FOR I=1 TO 2:LOCATE 23,1:PRINT"PAGE"I:NEXT:NEXT:RETURN
J 14000 SCREEN 0,0,0,7:CLS:PRINT"SCREEN 0,0,0,7":RETURN
P 14100 SCREEN 0,0,1,7:CLS:PRINT"SCREEN 0,0,1,7":RETURN
X 14200 IF K(1,15)=0 THEN K(1,15)=1:COLOR ,,15:PRINT"COLOR ,,15";:ELSE K(1,15)=0:COLOR ,,0:PRINT"COLOR ,,0";
F 14300 END
A 14400 BUREAU OF WEATHER STATISTICS
A 14500 COLOR 15,1:CLS:LOCATE 4,3:PRINT"BUREAU OF WEATHER STATISTICS":COLOR 7,0
G 14600 LOCATE 15,5:PRINT"CURRENT  CONDITIONS":LOCATE 17,5:PRINT"FORECAST  FOR  TOMORROW":LOCATE 19,5:PRINT"LONG  RANGE  FORECAST"
N 14700 LOCATE 15+2*(Y=1),3:PRINT"=>":LOCATE 15+2*(Y=2),3:PRINT"=>":LOCATE 15+2*(Y=3),3:PRINT"=>":RETURN
B 14800 LOCATE 15,5:PRINT"CURRENT  CONDITIONS":LOCATE 17,5:PRINT"FORECAST  FOR  TOMORROW":LOCATE 19,5:PRINT"LONG  RANGE  FORECAST"
W 14900 LOCATE 15+2*(Y=1),3:PRINT"=>":LOCATE 15+2*(Y=2),3:PRINT"=>":LOCATE 15+2*(Y=3),3:PRINT"=>":RETURN
V 15000 LOCATE 11,3:PRINT"WHEAT  IN  STORAGE:";BU;"BUSHELS":LOCATE 12,3:PRINT"WHEAT  PRICE:  $";PR;"PER  BUSHEL"
N 15100 LOCATE 25,1:PRINT"TAB(";:IF Y=1 THEN PRINT"BUY";:IF Y=2 THEN PRINT"SELL";:IF Y=3 THEN PRINT"EXIT";:PRINT")";:RETURN
B 15200 LOCATE 20,5:PRINT"SELECT  ONE:  ";:RETURN
N 15400 LOCATE 20,5:PRINT"SELECT  ONE:  ";:RETURN
N 15500 LOCATE 21,1:PRINT"HOW  MANY  SOLDIERS";:RETURN
W 15700 GOSUB 15500:LOCATE ,,1:GOSUB 12500:IF KEY$="" THEN 15700 ELSE IF KEY$="0" THEN RETURN ELSE IF KEY$="1" THEN SC=SC+2:RETURN ELSE IF KEY$="2" THEN SC=SC+4:RETURN ELSE 15700
P 15800 BKS$="":IF KEY$="" THEN 15800 ELSE IF KEY$="6" THEN 15800 ELSE BKS$=KEY$:PRINT KEY$;:RETURN
H 15900 IF KEY$="6" THEN LOCATE ,,0:RETURN
W 16000 VAL(KEY$)=VAL(KEY$)+1:RETURN
J 16100 IF VAL(KEY$)>15 THEN VAL(KEY$)=0
K 16200 COLOR VAL(KEY$):RETURN
S 16300 IF KEY$="" THEN 16300 ELSE IF KEY$="6" THEN RETURN ELSE IF KEY$="1" THEN SC=SC-2:RETURN ELSE IF KEY$="2" THEN SC=SC-4:RETURN ELSE 16300
N 16400 LOCATE 21,1:PRINT"HOW  MANY  SOLDIERS";:RETURN
I 16500 LOCATE ?(DEPOSIT),20:PRINT"HOW MUCH DEPOSIT";:RETURN
D 16600 LOCATE ?(WITHDRAW),20:PRINT"HOW MUCH WITHDRAW";:RETURN
R 16700 LOCATE 20,1:PRINT"HOW  MUCH  WHEAT  TO  BUY";:RETURN
N 16800 LOCATE 20,1:PRINT"HOW  MUCH  WHEAT  TO  SELL";:RETURN
A 16900 LOCATE 20,1:PRINT"HOW  MUCH  TO  INVEST";:RETURN
N 17000 LOCATE 20,1:PRINT"HOW  MUCH  TO  BORROW";:RETURN
N 17100 LOCATE 20,1:PRINT"HOW  MUCH  TO  REPAY";:RETURN
S 17200 IF KEY$="" THEN 17200 ELSE IF KEY$="6" THEN RETURN ELSE IF KEY$="1" THEN SC=SC-2:RETURN ELSE IF KEY$="2" THEN SC=SC-4:RETURN ELSE 17200
W 17600 PRINT"INPUT  MESSAGE":LOCATE 15,1:LINE INPUT MES$:RETURN
C 17700 PRINT"BAD  MESSAGE":RETURN
T 17800 FOR X=1 TO 15:COLOR X:PRINT MES$:NEXT:RETURN
D 17900 FOR X=1 TO 15:COLOR X:PRINT MES$:NEXT:RETURN
V 18000 RETURN
Q 18100 RETURN
S 18200 SOLDIERS  SCREEN
S 18300 FOR P1=4,7:COLOR ,,7:FOR D=1 TO 2:NEXT:NEXT:RETURN
N 18400 LOCATE 25:PRINT"SOLDIERS";:LOCATE 2:PRINT"HOW  MANY  SOLDIERS  TO  SEND";:RETURN
Y 18500 LOCATE 4,8:PRINT"SOLDIERS";:LOCATE 5,8:PRINT"SOLDIERS";:RETURN
Y 18600 LOCATE 7,8:PRINT"GOLD  PIECES";:LOCATE 8,8:PRINT"GOLD  PIECES";:RETURN
A 18700 LOCATE 13,11:PRINT"1)  HIRE  SOLDIERS":LOCATE 14,11:PRINT"2)  FIRE  SOLDIERS"
S 18800 LOCATE 15,11:PRINT"3)  ADJUST  PAY":LOCATE 16,11:PRINT"4)  EXIT"
Y 18900 LOCATE ,,0:RETURN
H 19000 LOCATE 6,24:PRINT"USING  A$";:LOCATE 7,24:PRINT"USING  A$";:RETURN
O 19100 LOCATE 21,8:PRINT"SELECT  ONE:  ";:RETURN
K 19200 RETURN
N 19300 RESOLDIER  OPTION
P 19400 SCREEN 0,,,1:GOSUB 19500:SCREEN 0,,,0:RETURN
A 19500 IF KEY$="" THEN 19500 ELSE IF KEY$="4" THEN 19590 ELSE IF KEY$="1" OR KEY$="2" OR KEY$="3" THEN 19600 ELSE 19500
C 19600 IF KEY$="4" THEN LOCATE ,,0:RETURN
F 19700 IF KEY$="1" THEN BUY=BUY+VAL(KEY$):GOSUB 19900:RETURN ELSE IF KEY$="2" THEN BUY=BUY-VAL(KEY$):GOSUB 19900:RETURN ELSE IF KEY$="3" THEN 19590 ELSE 19500
Y 19800 LOCATE 18,19:PRINT"WHEAT  IN  STORAGE:";BU;"BUSHELS":LOCATE 19,19:PRINT"WHEAT  PRICE:  $";PR;"PER  BUSHEL":RETURN
G 19900 LOCATE 21,1:PRINT"HOW  MANY  BUSHELS  TO  BUY";:RETURN
L 20000 LOCATE 29,1:PRINT"HOW  MANY  BUSHELS  TO  SELL";:RETURN
Y 20000 LOCATE 29,1:PRINT"HOW  MANY  BUSHELS  TO  SELL";:RETURN
R 20100 RETURN
N 20200 LOCATE 21,1:PRINT"HOW  MANY  SOLDIERS";:RETURN
M 20300 LOCATE 21,1:PRINT"HOW  MANY  SOLDIERS";:RETURN
U 20400 LOCATE 21,1:PRINT"SOLDIERS";:RETURN
U 20500 LOCATE 21,1:PRINT"SOLDIERS";:RETURN
A 20600 RETURN
P 20700 RETURN
S 20800 RETURN
P 20900 RETURN
B 21000 RETURN
X 21100 LOCATE 2,8:PRINT"SOLDIERS";:RETURN
M 21200 LOCATE 5,8:PRINT"COMMERCE  TAX";:RETURN
```

Continued



HCM



Continued

```
100 REM SERF CITY
110 REM BY JIM PETERSON
120 REM COPYRIGHT 1985
130 REM HOME COMPUTER MAGAZINE
140 CALL CLEAR
150 CALL SCREEN(2)
160 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
170 CALL CHAR(97,"0000000000000000")
180 CALL CHAR(98,"0000000000000000")
190 CALL CHAR(99,"0000000000000000")
200 CALL CHAR(100,"0000000000000000")
210 CALL CHAR(101,"0000000000000000")
220 CALL CHAR(102,"0000000000000000")
230 CALL CHAR(103,"0000000000000000")
240 CALL CHAR(104,"0000000000000000")
250 CALL CHAR(105,"0000000000000000")
260 CALL CHAR(106,"0000000000000000")
270 CALL CHAR(107,"0000000000000000")
280 CALL CHAR(108,"0000000000000000")
290 CALL CHAR(109,"0000000000000000")
300 CALL CHAR(110,"0000000000000000")
310 CALL CHAR(111,"0000000000000000")
320 CALL CHAR(112,"0000000000000000")
330 CALL CHAR(113,"0000000000000000")
340 CALL CHAR(114,"0000000000000000")
350 CALL CHAR(115,"0000000000000000")
360 CALL CHAR(116,"0000000000000000")
370 CALL CHAR(117,"0000000000000000")
380 CALL CHAR(118,"0000000000000000")
390 CALL CHAR(119,"0000000000000000")
400 CALL CHAR(120,"0000000000000000")
410 CALL CHAR(121,"0000000000000000")
420 CALL CHAR(122,"0000000000000000")
430 CALL CHAR(123,"0000000000000000")
440 CALL CHAR(124,"0000000000000000")
450 CALL CHAR(125,"0000000000000000")
460 CALL CHAR(126,"0000000000000000")
470 CALL CHAR(127,"0000000000000000")
480 CALL CHAR(128,"0000000000000000")
490 CALL CHAR(129,"0000000000000000")
500 CALL CHAR(130,"0000000000000000")
510 CALL CHAR(131,"0000000000000000")
520 CALL CHAR(132,"0000000000000000")
530 CALL CHAR(133,"0000000000000000")
540 CALL CHAR(134,"0000000000000000")
550 CALL CHAR(135,"0000000000000000")
560 CALL CHAR(136,"0000000000000000")
570 CALL CHAR(137,"0000000000000000")
580 CALL CHAR(138,"0000000000000000")
590 CALL CHAR(139,"0000000000000000")
600 CALL CHAR(140,"0000000000000000")
610 CALL CHAR(141,"0000000000000000")
620 CALL CHAR(142,"0000000000000000")
630 CALL CHAR(143,"0000000000000000")
640 CALL CHAR(144,"0000000000000000")
650 CALL CHAR(145,"0000000000000000")
660 CALL CHAR(146,"0000000000000000")
670 CALL CHAR(147,"0000000000000000")
680 CALL CHAR(148,"0000000000000000")
690 CALL CHAR(149,"0000000000000000")
700 CALL CHAR(150,"0000000000000000")
710 CALL CHAR(151,"0000000000000000")
720 CALL CHAR(152,"0000000000000000")
730 CALL CHAR(153,"0000000000000000")
740 CALL CHAR(154,"0000000000000000")
750 CALL CHAR(155,"0000000000000000")
760 CALL CHAR(156,"0000000000000000")
770 CALL CHAR(157,"0000000000000000")
780 CALL CHAR(158,"0000000000000000")
790 CALL CHAR(159,"0000000000000000")
800 CALL CHAR(160,"0000000000000000")
810 CALL CHAR(161,"0000000000000000")
820 CALL CHAR(162,"0000000000000000")
830 CALL CHAR(163,"0000000000000000")
840 CALL CHAR(164,"0000000000000000")
850 CALL CHAR(165,"0000000000000000")
860 CALL CHAR(166,"0000000000000000")
870 CALL CHAR(167,"0000000000000000")
880 CALL CHAR(168,"0000000000000000")
890 CALL CHAR(169,"0000000000000000")
900 CALL CHAR(170,"0000000000000000")
910 CALL CHAR(171,"0000000000000000")
920 CALL CHAR(172,"0000000000000000")
930 CALL CHAR(173,"0000000000000000")
940 CALL CHAR(174,"0000000000000000")
950 CALL CHAR(175,"0000000000000000")
960 CALL CHAR(176,"0000000000000000")
970 CALL CHAR(177,"0000000000000000")
980 CALL CHAR(178,"0000000000000000")
990 CALL CHAR(179,"0000000000000000")
1000 CALL CHAR(180,"0000000000000000")
1010 CALL CHAR(181,"0000000000000000")
1020 CALL CHAR(182,"0000000000000000")
1030 CALL CHAR(183,"0000000000000000")
1040 CALL CHAR(184,"0000000000000000")
1050 CALL CHAR(185,"0000000000000000")
1060 CALL CHAR(186,"0000000000000000")
1070 CALL CHAR(187,"0000000000000000")
1080 CALL CHAR(188,"0000000000000000")
1090 CALL CHAR(189,"0000000000000000")
1100 CALL CHAR(190,"0000000000000000")
1110 CALL CHAR(191,"0000000000000000")
1120 CALL CHAR(192,"0000000000000000")
1130 CALL CHAR(193,"0000000000000000")
1140 CALL CHAR(194,"0000000000000000")
1150 CALL CHAR(195,"0000000000000000")
1160 CALL CHAR(196,"0000000000000000")
1170 CALL CHAR(197,"0000000000000000")
1180 CALL CHAR(198,"0000000000000000")
1190 CALL CHAR(199,"0000000000000000")
1200 CALL CHAR(200,"0000000000000000")
1210 CALL CHAR(201,"0000000000000000")
1220 CALL CHAR(202,"0000000000000000")
1230 CALL CHAR(203,"0000000000000000")
1240 CALL CHAR(204,"0000000000000000")
1250 CALL CHAR(205,"0000000000000000")
1260 CALL CHAR(206,"0000000000000000")
1270 CALL CHAR(207,"0000000000000000")
1280 CALL CHAR(208,"0000000000000000")
1290 CALL CHAR(209,"0000000000000000")
1300 CALL CHAR(210,"0000000000000000")
1310 CALL CHAR(211,"0000000000000000")
1320 CALL CHAR(212,"0000000000000000")
1330 CALL CHAR(213,"0000000000000000")
1340 CALL CHAR(214,"0000000000000000")
1350 CALL CHAR(215,"0000000000000000")
1360 CALL CHAR(216,"0000000000000000")
1370 CALL CHAR(217,"0000000000000000")
1380 CALL CHAR(218,"0000000000000000")
1390 CALL CHAR(219,"0000000000000000")
1400 CALL CHAR(220,"0000000000000000")
1410 CALL CHAR(221,"0000000000000000")
1420 CALL CHAR(222,"0000000000000000")
1430 CALL CHAR(223,"0000000000000000")
1440 CALL CHAR(224,"0000000000000000")
1450 CALL CHAR(225,"0000000000000000")
1460 CALL CHAR(226,"0000000000000000")
1470 CALL CHAR(227,"0000000000000000")
1480 CALL CHAR(228,"0000000000000000")
1490 CALL CHAR(229,"0000000000000000")
1500 CALL CHAR(230,"0000000000000000")
1510 CALL CHAR(231,"0000000000000000")
1520 CALL CHAR(232,"0000000000000000")
1530 CALL CHAR(233,"0000000000000000")
1540 CALL CHAR(234,"0000000000000000")
1550 CALL CHAR(235,"0000000000000000")
1560 CALL CHAR(236,"0000000000000000")
1570 CALL CHAR(237,"0000000000000000")
1580 CALL CHAR(238,"0000000000000000")
1590 CALL CHAR(239,"0000000000000000")
1600 CALL CHAR(240,"0000000000000000")
1610 CALL CHAR(241,"0000000000000000")
1620 CALL CHAR(242,"0000000000000000")
1630 CALL CHAR(243,"0000000000000000")
1640 CALL CHAR(244,"0000000000000000")
1650 CALL CHAR(245,"0000000000000000")
1660 CALL CHAR(246,"0000000000000000")
1670 CALL CHAR(247,"0000000000000000")
1680 CALL CHAR(248,"0000000000000000")
1690 CALL CHAR(249,"0000000000000000")
1700 CALL CHAR(250,"0000000000000000")
1710 CALL CHAR(251,"0000000000000000")
1720 CALL CHAR(252,"0000000000000000")
1730 CALL CHAR(253,"0000000000000000")
1740 CALL CHAR(254,"0000000000000000")
1750 CALL CHAR(255,"0000000000000000")
1760 CALL CHAR(32,"0000000000000000")
1770 CALL CHAR(33,"0000000000000000")
1780 CALL CHAR(34,"0000000000000000")
1790 CALL CHAR(35,"0000000000000000")
1800 CALL CHAR(36,"0000000000000000")
1810 CALL CHAR(37,"0000000000000000")
1820 CALL CHAR(38,"0000000000000000")
1830 CALL CHAR(39,"0000000000000000")
1840 CALL CHAR(40,"0000000000000000")
1850 CALL CHAR(41,"0000000000000000")
1860 CALL CHAR(42,"0000000000000000")
1870 CALL CHAR(43,"0000000000000000")
1880 CALL CHAR(44,"0000000000000000")
1890 CALL CHAR(45,"0000000000000000")
1900 CALL CHAR(46,"0000000000000000")
1910 CALL CHAR(47,"0000000000000000")
1920 CALL CHAR(48,"0000000000000000")
1930 CALL CHAR(49,"0000000000000000")
1940 CALL CHAR(50,"0000000000000000")
1950 CALL CHAR(51,"0000000000000000")
1960 CALL CHAR(52,"0000000000000000")
1970 CALL CHAR(53,"0000000000000000")
1980 CALL CHAR(54,"0000000000000000")
1990 CALL CHAR(55,"0000000000000000")
2000 CALL CHAR(56,"0000000000000000")
2010 CALL CHAR(57,"0000000000000000")
2020 CALL CHAR(58,"0000000000000000")
2030 CALL CHAR(59,"0000000000000000")
2040 CALL CHAR(60,"0000000000000000")
2050 CALL CHAR(61,"0000000000000000")
2060 CALL CHAR(62,"0000000000000000")
2070 CALL CHAR(63,"0000000000000000")
2080 CALL CHAR(64,"0000000000000000")
2090 CALL CHAR(65,"0000000000000000")
2100 CALL CHAR(66,"0000000000000000")
2110 CALL CHAR(67,"0000000000000000")
2120 CALL CHAR(68,"0000000000000000")
2130 CALL CHAR(69,"0000000000000000")
2140 CALL CHAR(70,"0000000000000000")
2150 CALL CHAR(71,"0000000000000000")
2160 CALL CHAR(72,"0000000000000000")
2170 CALL CHAR(73,"0000000000000000")
2180 CALL CHAR(74,"0000000000000000")
2190 CALL CHAR(75,"0000000000000000")
2200 CALL CHAR(76,"0000000000000000")
2210 CALL CHAR(77,"0000000000000000")
2220 CALL CHAR(78,"0000000000000000")
2230 CALL CHAR(79,"0000000000000000")
2240 CALL CHAR(80,"0000000000000000")
2250 CALL CHAR(81,"0000000000000000")
2260 CALL CHAR(82,"0000000000000000")
2270 CALL CHAR(83,"0000000000000000")
2280 CALL CHAR(84,"0000000000000000")
2290 CALL CHAR(85,"0000000000000000")
2300 CALL CHAR(86,"0000000000000000")
2310 CALL CHAR(87,"0000000000000000")
2320 CALL CHAR(88,"0000000000000000")
2330 CALL CHAR(89,"0000000000000000")
2340 CALL CHAR(90,"0000000000000000")
2350 CALL CHAR(91,"0000000000000000")
2360 CALL CHAR(92,"0000000000000000")
2370 CALL CHAR(93,"0000000000000000")
2380 CALL CHAR(94,"0000000000000000")
2390 CALL CHAR(95,"0000000000000000")
2400 CALL CHAR(96,"0000000000000000")
2410 CALL CHAR(97,"0000000000000000")
2420 CALL CHAR(98,"0000000000000000")
2430 CALL CHAR(99,"0000000000000000")
2440 CALL CHAR(100,"0000000000000000")
2450 CALL CHAR(101,"0000000000000000")
2460 CALL CHAR(102,"0000000000000000")
2470 CALL CHAR(103,"0000000000000000")
2480 CALL CHAR(104,"0000000000000000")
2490 CALL CHAR(105,"0000000000000000")
2500 CALL CHAR(106,"0000000000000000")
2510 CALL CHAR(107,"0000000000000000")
2520 CALL CHAR(108,"0000000000000000")
2530 CALL CHAR(109,"0000000000000000")
2540 CALL CHAR(110,"0000000000000000")
2550 CALL CHAR(111,"0000000000000000")
2560 CALL CHAR(112,"0000000000000000")
2570 CALL CHAR(113,"0000000000000000")
2580 CALL CHAR(114,"0000000000000000")
2590 CALL CHAR(115,"0000000000000000")
2600 CALL CHAR(116,"0000000000000000")
2610 CALL CHAR(117,"0000000000000000")
2620 CALL CHAR(118,"0000000000000000")
2630 CALL CHAR(119,"0000000000000000")
2640 CALL CHAR(120,"0000000000000000")
2650 CALL CHAR(121,"0000000000000000")
2660 CALL CHAR(122,"0000000000000000")
2670 CALL CHAR(123,"0000000000000000")
2680 CALL CHAR(124,"0000000000000000")
2690 CALL CHAR(125,"0000000000000000")
2700 CALL CHAR(126,"0000000000000000")
2710 CALL CHAR(127,"0000000000000000")
2720 CALL CHAR(128,"0000000000000000")
2730 CALL CHAR(129,"0000000000000000")
2740 CALL CHAR(130,"0000000000000000")
2750 CALL CHAR(131,"0000000000000000")
2760 CALL CHAR(132,"0000000000000000")
2770 CALL CHAR(133,"0000000000000000")
2780 CALL CHAR(134,"0000000000000000")
2790 CALL CHAR(135,"0000000000000000")
2800 CALL CHAR(136,"0000000000000000")
2810 CALL CHAR(137,"0000000000000000")
2820 CALL CHAR(138,"0000000000000000")
2830 CALL CHAR(139,"0000000000000000")
2840 CALL CHAR(140,"0000000000000000")
2850 CALL CHAR(141,"0000000000000000")
2860 CALL CHAR(142,"0000000000000000")
2870 CALL CHAR(143,"0000000000000000")
2880 CALL CHAR(144,"0000000000000000")
2890 CALL CHAR(145,"0000000000000000")
2900 CALL CHAR(146,"0000000000000000")
2910 CALL CHAR(147,"0000000000000000")
2920 CALL CHAR(148,"0000000000000000")
2930 CALL CHAR(149,"0000000000000000")
2940 CALL CHAR(150,"0000000000000000")
2950 CALL CHAR(151,"0000000000000000")
2960 CALL CHAR(152,"0000000000000000")
2970 CALL CHAR(153,"0000000000000000")
2980 CALL CHAR(154,"0000000000000000")
2990 CALL CHAR(155,"0000000000000000")
3000 CALL CHAR(156,"0000000000000000")
3010 CALL CHAR(157,"0000000000000000")
3020 CALL CHAR(158,"0000000000000000")
3030 CALL CHAR(159,"0000000000000000")
3040 CALL CHAR(160,"0000000000000000")
3050 CALL CHAR(161,"0000000000000000")
3060 CALL CHAR(162,"0000000000000000")
3070 CALL CHAR(163,"0000000000000000")
3080 CALL CHAR(164,"0000000000000000")
3090 CALL CHAR(165,"0000000000000000")
3100 CALL CHAR(166,"0000000000000000")
3110 CALL CHAR(167,"0000000000000000")
3120 CALL CHAR(168,"0000000000000000")
3130 CALL CHAR(169,"0000000000000000")
3140 CALL CHAR(170,"0000000000000000")
3150 CALL CHAR(171,"0000000000000000")
3160 CALL CHAR(172,"0000000000000000")
3170 CALL CHAR(173,"0000000000000000")
3180 CALL CHAR(174,"0000000000000000")
3190 CALL CHAR(175,"0000000000000000")
3200 CALL CHAR(176,"0000000000000000")
3210 CALL CHAR(177,"0000000000000000")
3220 CALL CHAR(178,"0000000000000000")
3230 CALL CHAR(179,"0000000000000000")
3240 CALL CHAR(180,"0000000000000000")
3250 CALL CHAR(181,"0000000000000000")
3260 CALL CHAR(182,"0000000000000000")
3270 CALL CHAR(183,"0000000000000000")
3280 CALL CHAR(184,"0000000000000000")
3290 CALL CHAR(185,"0000000000000000")
3300 CALL CHAR(186,"0000000000000000")
3310 CALL CHAR(187,"0000000000000000")
3320 CALL CHAR(188,"0000000000000000")
3330 CALL CHAR(189,"0000000000000000")
3340 CALL CHAR(190,"0000000000000000")
3350 CALL CHAR(191,"0000000000000000")
3360 CALL CHAR(192,"0000000000000000")
3370 CALL CHAR(193,"0000000000000000")
3380 CALL CHAR(194,"0000000000000000")
3390 CALL CHAR(195,"0000000000000000")
3400 CALL CHAR(196,"0000000000000000")
3410 CALL CHAR(197,"0000000000000000")
3420 CALL CHAR(198,"0000000000000000")
3430 CALL CHAR(199,"0000000000000000")
3440 CALL CHAR(200,"0000000000000000")
3450 CALL CHAR(201,"0000000000000000")
3460 CALL CHAR(202,"0000000000000000")
3470 CALL CHAR(203,"0000000000000000")
3480 CALL CHAR(204,"0000000000000000")
3490 CALL CHAR(205,"0000000000000000")
3500 CALL CHAR(206,"0000000000000000")
3510 CALL CHAR(207,"0000000000000000")
3520 CALL CHAR(208,"0000000000000000")
3530 CALL CHAR(209,"0000000000000000")
3540 CALL CHAR(210,"0000000000000000")
3550 CALL CHAR(211,"0000000000000000")
3560 CALL CHAR(212,"0000000000000000")
3570 CALL CHAR(213,"0000000000000000")
3580 CALL CHAR(214,"0000000000000000")
3590 CALL CHAR(215,"0000000000000000")
3600 CALL CHAR(216,"0000000000000000")
3610 CALL CHAR(217,"0000000000000000")
3620 CALL CHAR(218,"0000000000000000")
3630 CALL CHAR(219,"0000000000000000")
3640 CALL CHAR(220,"0000000000000000")
3650 CALL CHAR(221,"0000000000000000")
3660 CALL CHAR(222,"0000000000000000")
3670 CALL CHAR(223,"0000000000000000")
3680 CALL CHAR(224,"0000000000000000")
3690 CALL CHAR(225,"0000000000000000")
3700 CALL CHAR(226,"0000000000000000")
3710 CALL CHAR(227,"0000000000000000")
3720 CALL CHAR(228,"0000000000000000")
3730 CALL CHAR(229,"0000000000000000")
3740 CALL CHAR(230,"0000000000000000")
3750 CALL CHAR(231,"0000000000000000")
3760 CALL CHAR(232,"0000000000000000")
3770 CALL CHAR(233,"0000000000000000")
3780 CALL CHAR(234,"0000000000000000")
3790 CALL CHAR(235,"0000000000000000")
3800 CALL CHAR(236,"0000000000000000")
3810 CALL CHAR(237,"0000000000000000")
3820 CALL CHAR(238,"0000000000000000")
3830 CALL CHAR(239,"0000000000000000")
3840 CALL CHAR(240,"0000000000000000")
3850 CALL CHAR(241,"0000000000000000")
3860 CALL CHAR(242,"0000000000000000")
3870 CALL CHAR(243,"0000000000000000")
3880 CALL CHAR(244,"0000000000000000")
3890 CALL CHAR(245,"0000000000000000")
3900 CALL CHAR(246,"0000000000000000")
3910 CALL CHAR(247,"0000000000000000")
3920 CALL CHAR(248,"0000000000000000")
3930 CALL CHAR(249,"0000000000000000")
```

Continued

TYPE-IN LISTINGS

SERF CITY *Continued*


```
A 3940  PRINT #KP(P1,A3):"PLEASE ENTER LANTABLE LANDED:";TA
B       P(16):#KP("F"D"GOLD PIECES:";TAB(16);K(P1,12
T 3950  PRINT #KP("F"D"WOOPIE PIECES:";TAB(16);K(P1,A8
N 3960  PRINT #KP("F"D"WHOOPIE PIECES:";TAB(16);K(P1,A9
J 3970  PRINT #KP("F"D"WHOLE PIECES:";TAB(16);K(P1,10
J 3980  PRINT #KP("F"D"PERSONAL TAX:";TAB(16);K(P1,14
N 3990  PRINT #KP("COMMERCIAL TAX:";TAB(16);K(P1,15
V 4000  PRINT #KP("WHEAT":"SOLDI
E 4010  PRINT "PRESS ENTER
UK4020  RETURN
CW4030  CALL CLEAR
WR4040  CALL SCREEN(A1 TO 23
RD4050  PRINT #Z:S$(Z)
DC4060  PRINT #Z:S$(24);
CI4070  RETURN
LF4110  NEXT Z
RE4120  END
```

```
A 4130  FOR Z=A8 TO 11
L 4140  REXT Z
T 4150  RNS$(21)="                                "
N 4160  RPSS$(23)="MOVANTE WITH ESSERT"
C 4170  RSS$(23)="ENDER TO EXIT"
Y 4180  RSS$(24)="ENTER A$"
S 4190  FOR Z=A1 TO 8
E 4200  RA$(Z)=A$
Z 4210  RCALL CHAR(A,A$)
D 4220  NCALL CHAR(A9,15,13)
S 4230  CALL COLOR(10,15,14)
K 4240  CALL COLOR(11,A7,12)
D 4250  CALL CHAR(120,"
F 4260  RCALL CHAR(121,"
L 4270  NCALL CHAR(122,"
M 4280  DATA 
        ,
        ,
D 4290  DDATA 
R 4300  DATA 
        ,
L 4310  DATA 
V 4320  CFOR Z=A1 TO 5
Z 4330  NEXT Z
N 4340  RCALL KEY(A0,W,S)
F 4350  IF S=A0 THEN 4350
W 4360  RETURN
U 4370  
```

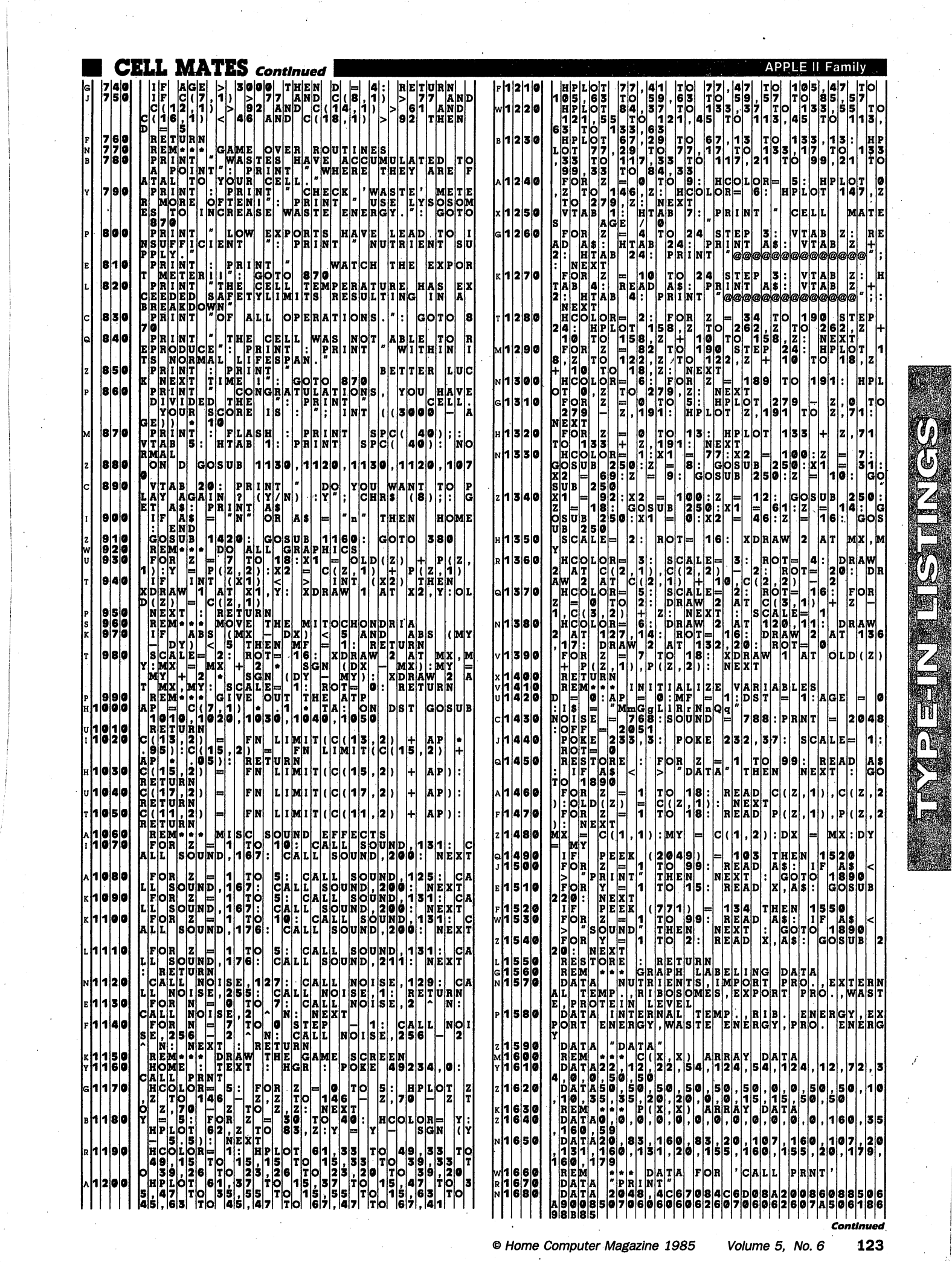
CELL MATES


```
F 100   REM
G 110   REM
J 120   REM
Y 130   REM
N 140   REM
H 150   REM
F 160   REM
A 170   REM
N 180   REM
G 190   REM
Q 200   REM
Z 210   REM
C 220   REM
Z 230   REM
A 240   REM
M 250   REM
Z 260   REM
C 270   REM
G 280   REM
G 290   REM
P 300   REM
A 310   REM
H 320   REM
P 330   REM
A 340   REM
K 350   REM
A 360   REM
N 370   REM
L 380   REM
L 390   REM
Y 400   REM
P 410   REM
Y 420   REM
M 430   REM
Q 440   REM
M 450   REM
S 460   REM
  470   REM
```

```
D 480   IF A$="Y" THEN 
N       IF A$="N" THEN 
H       GOSUB 
A 490   RETURN
W 500   REM
N 510   REM
        
        
        
        
        
        
        
V 520   
        
Q 530   
        
Z 540   
        
        
K 550   
        
V 560   
        
E 570   
        
X 580   
        
X 590   
        
N 600   
I 610   
        
V 620   
        
P 630   
P 640   
        
M 650   
O 660   
S 670   
        
R 680   
        
Q 690   
        
        
I 700   
Y 710   
        
X 720   
I 730   
```

Continued

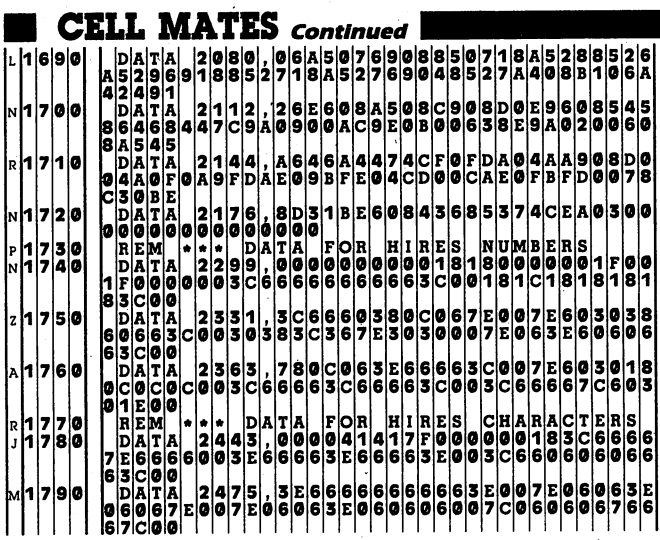
CELL MATES *Continued*

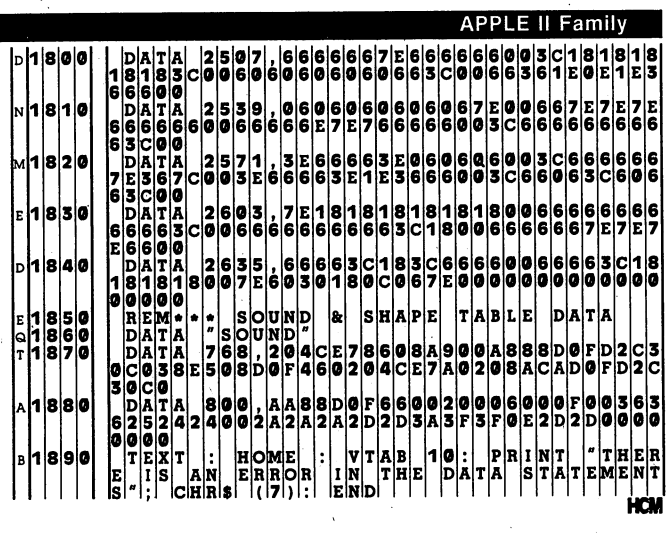


Continued

CELL MATES *Continued*

APPLE II Family

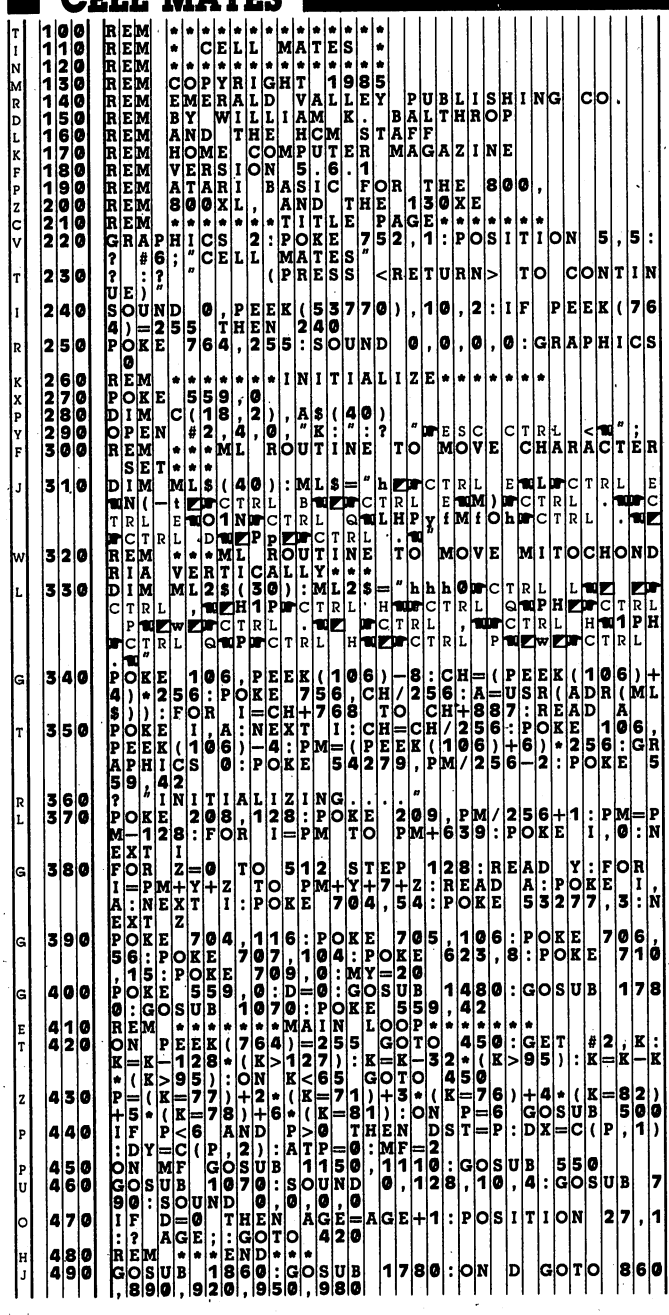


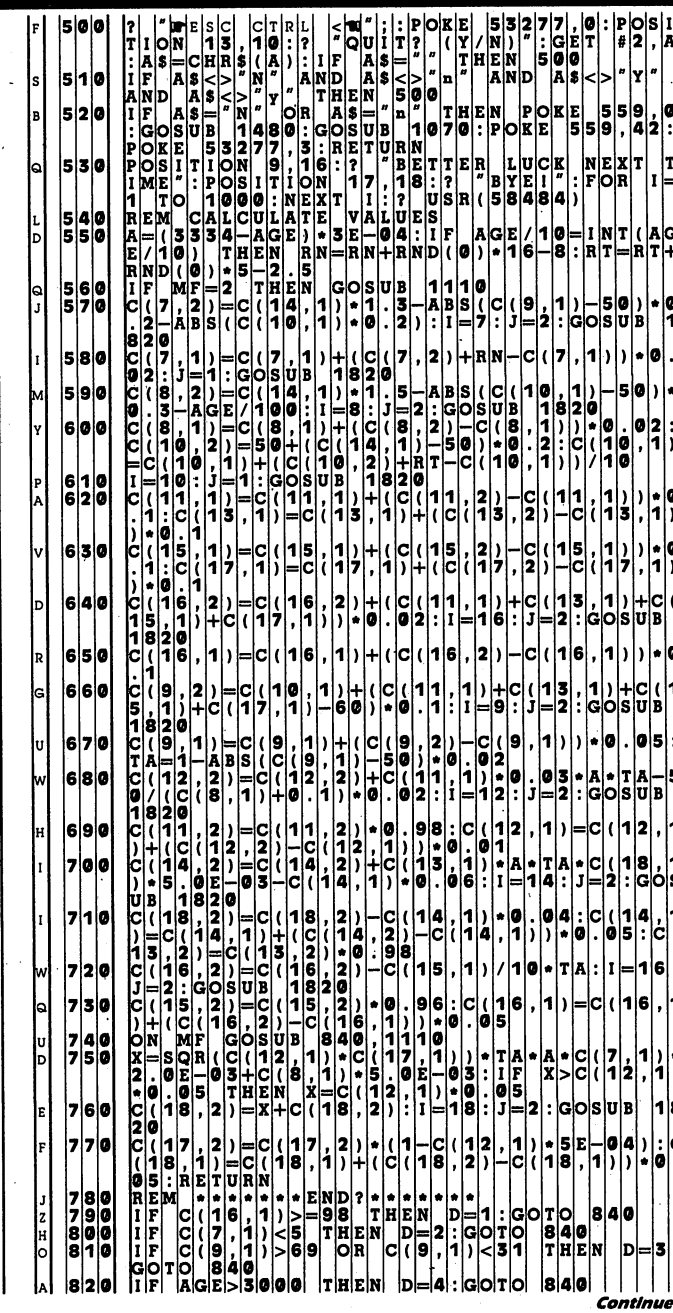


HCM

CELL MATES

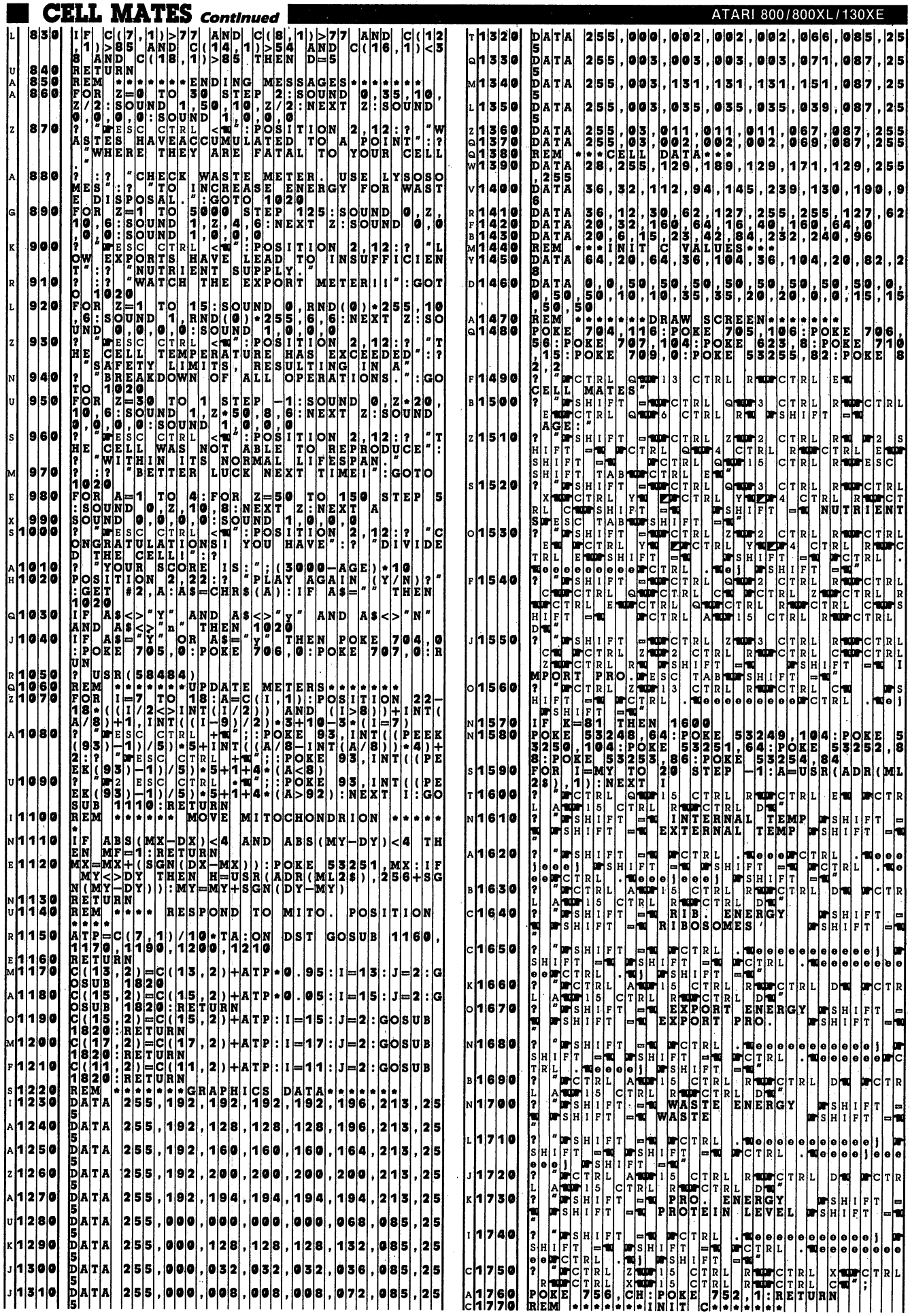
ATARI 800/800XL/130XE





Continued

CELL MATES *Continued*



Continued

CELL MATES *Continued* — ATARI 800/800XL/130XE

```
W 1780 MX=64:DX=MX:MY=20:DY=MY:RESTORE 1450:FOR I=1 TO 18:READ A,Z:C(I,1)=A:C(I,2)=Z:NEXT I:AGE=0:RETURN
H 1790 REM *** MIN & MAX ***
K 1800 REM SEND I & J
M 1810 REM OUTPUT: I=MIN(I,J) J=MAX(I,J)
Q 1820 IF C(I,J)<0 THEN C(I,J)=0:RETURN
Z 1830 IF C(I,J)>100 THEN C(I,J)=100:RETURN
J 1840 RETURN
P 1850 REM ******DEATH OF CELL******
E 1860 FOR I=48 TO 60:POKE 53248,I:POKE 53249,108-I:POKE 53250,I/2+48:POKE 53251,I/4+56:NEXT I:ON RND(0) GOTO 1860
R 1870 IF PEEK(704)>15 THEN POKE 704,15:SOUND 0,I,0,4:GOTO 1860
T 1880 IF PEEK(705)>15 THEN POKE 705,15:SOUND 0,I,0,5:GOTO 1860
V 1890 IF PEEK(706)>15 THEN POKE 706,15:SOUND 0,I,0,6:GOTO 1860
I 1900 IF PEEK(707)>15 THEN POKE 707,15:SOUND 0,I,0,7:GOTO 1860
U 1910 FOR I=PEEK(711) TO 14 STEP -1:POKE 711,I:SOUND 0,I,0,8:NEXT I
J 1920 SOUND 0,0,0,0:RETURN
```

HCM

CELL MATES — COMMODORE 64

```
T 100 REM ***************
I 110 REM * CELL MATES *
N 120 REM ***************
M 130 REM COPYRIGHT 1985
O 140 REM EMERALD VALLEY PUBLIBHING CO.
D 150 REM BY WILLIAM K. BALTHROP
L 160 REM AND THE HCM STAFF
K 170 REM HOME COMPUTER MAGAZINE
F 180 REM VERSION 5.6.1
Y 190 REM C-64 BASIC
M 200 REM
I 210 POKE 52,48:POKE 56,48:CLR
Y 220 DIM C(19,2):DIM METR(13):DIM MHST(13,2):DIM KEYS(5)
V 230 DEF FN MN(X) = X+(X>100)*(X-100):DEF FN MX(X) = X+(X<0)*X
K 240 CBASE=12288:SBASE=14336:MCHARS=128:SND=54272:VID=1024:QI=0
G 250 BRDR=4:BCKGR=0:CCOL=12:POKE53281,BCKGR:POKE53280,BRDR:PRINT CHR$(152)
H 260 PRINT"{SHIFT CLR}{CRSRRIGHT}{7 CRSR DOWN}{9 CRSRRIGHT}C E L L  M A T E S"
N 270 PRINT"{3 CRSRDOWN}{9 CRSRRIGHT}PRESS A KEY TO GO ON"
N 280 POKE 198,0:WAIT 198,1
Q 290 GOSUB1180:POKE 198,0
R 300 IF PEEK(198)<>0 THEN GOSUB790:IF MIT<0 THEN GOTO340
H 310 ON (-(MIT>0)+1) GOSUB690,640
T 320 GOSUB360:GOSUB610:GOSUB900:IF D=0 THEN GOSUB750:GOTO300
F 330 IF D=6 THEN GOSUB1930: IF D=0 THEN GOSUB750: GOTO300
B 340 POKE 53269,0:IF MIT>=0 THEN GOSUB970:IF D=0 THEN QI=1:RESTORE:AGE=0:GOTO290
N 350 PRINT"{SHIFT CLR}":END
Y 360 A=(3334-AGE)*.0003
Q 370 IF AGE*.1=INT(AGE*.1)THEN RN=RN+RND(0)*16-8:RT=RT+RND(0)*5-2.5
N 380 C(7,2)=FN MX(FN MN(C(14,1)*1.3-ABS(C(9,1)-50)*.2-ABS(C(10,1)-50)*.2))
A 390 C(7,1)=FN MX(FN MN(C(7,1)+(C(7,2)+RN-C(7,1))*.02))
T 400 C(8,2)=FN MX(FN MN(C(14,1)*1.5-ABS(C(10,1)-50)*.3-AGE*.01))
T 410 C(8,1)=C(8,1)+(C(8,2)-C(8,1))*.02:C(10,2)=50+(C(14,1)-50)*.2
R 420 C(10,1)=FN MX(FN MN(C(10,1)+(C(10,2)+RT-C(10,1))*.1))
L 430 C(11,1)=C(11,1)+(C(11,2)-C(11,1))*.1:C(13,1)=C(13,1)+(C(13,2)-C(13,1))*.1
E 440 C(15,1)=C(15,1)+(C(15,2)-C(15,1))*.1:C(17,1)=C(17,1)+(C(17,2)-C(17,1))*.1
V 450 C(16,2)=FN MX(FN MN(C(16,2)+(C(11,1)+C(13,1)+C(15,1)+C(17,1))*.02))
W 460 C(16,1)=C(16,1)+(C(16,2)-C(16,1))*.1
O 470 C(9,2)=FN MX(FN MN(C(10,1)+(C(11,1)+C(13,1)+C(15,1)+C(17,1)-60)*.2)*A)
X 480 C(9,1)=C(9,1)+(C(9,2)-C(9,1))*.05:TA=1-ABS(C(9,1)-50)*.02
A 490 C(12,2)=FN MX(FN MN(C(12,2)+C(11,1)*.03*A*TA-50/(C(8,1)+.1)*.02))
R 500 C(11,2)=C(11,2)*.98:C(12,1)=C(12,1)+(C(12,2)-C(12,1))*.01
T 510 C(14,2)=FN MX(FN MN(C(14,2)+C(13,1)*A*TA*C(18,1)*.005-C(14,1)*.15))
N 520 C(18,2)=C(18,2)-C(14,1)*.04:C(14,1)=C(14,1)+(C(14,2)-C(14,1))*.05
Y 530 C(13,2)=C(13,2)*.98
C 540 C(16,2)=FN MX(FN MN(C(16,2)-C(15,1)*.1*TA)):C(15,2)=C(15,2)*.98
C 550 C(16,1)=C(16,1)+(C(16,2)-C(16,1))*.05
N 560 Z=SQR(C(12,1)*C(17,1))*TA*A*C(7,1)*.002+C(8,1)*.005
O 570 IF (C(12,1)*.05)<Z THEN C(18,2)=FN MX(FN MN(C(12,1)*.05+C(18,2))):GOTO590
X 580 C(18,2)=FN MX(FN MN(Z+C(18,2)))
H 590 C(17,2)=C(17,2)*(1-C(12,1)*.0005):C(18,1)=C(18,1)+(C(18,2)-C(18,1))*.05
O 600 RETURN
A 610 FOR I=1 TO 12:RS=INT((C(I+6,1)*127)/100):RP=INT(RS/8):RS=RS-8*RP
R 620 BASE=MCHARS+(-9*(RP=0))+(-18*(RP=15))+1:POKE MHST(I,1)+METR(I),MHST(I,2)
G 630 MHST(I,1)=RP:MHST(I,2)=BASE-1:POKE METR(I)+RP,BASE+RS:NEXT I:RETURN
C 640 FOR I=1TO11:C(0,1)=C(0,1)+(C(0,1)>C(MIT,1))+ABS(C(0,1)<C(MIT,1))
Q 650 C(0,2)=C(0,2)+(C(0,2)>C(MIT,2))+ABS(C(0,2)<C(MIT,2))
F 660 POKE 53248,C(0,1):POKE 53249,C(0,2):NEXT
F 670 IF C(0,1)=C(MIT,1) AND C(0,2)=C(MIT,2) THEN C(6,1)=MIT:MIT=0
C 680 RETURN
C 690 IF C(6,1)=1 THEN RETURN
N 700 ATP=C(7,1)*.1*TA:ON C(6,1)-1 GOSUB710,720,730,740:RETURN
X 710 C(13,2)=FN MN(C(13,2)+ATP*.95):C(15,2)=FN MN(C(15,2)+ATP*.05):RETURN
E 720 C(15,2)=FN MN(C(15,2)+ATP):RETURN
F 730 C(17,2)=FN MN(C(17,2)+ATP):RETURN
U 740 C(11,2)=FN MN(C(11,2)+ATP):RETURN
W 750 AGE=AGE+1:A$=STR$(AGE)
F 760 POKE SND,10:POKESND+1,55:POKESND+6,255:POKESND+24,15:POKESND+4,33
S 770 FOR I=1 TO 10:NEXT:POKESND+4,32:POKE SND+24,0
Q 780 FOR I=1 TO LEN(A$):POKE 1131+I,ASC(MID$(A$,I,1)):NEXT I:RETURN
J 790 K=FN MX(PEEK(631)-75):POKE 198,0:IF K=6 THEN D=6:RETURN
T 800 IF K>4 THEN K=K-2
U 810 IF K<6 THEN MIT=KEYS(K):IF MIT>0 THEN GOSUB830
K 820 RETURN
R 830 POKE SND+24,15:POKE SND+5,50
O 840 FOR I=1 TO 10:POKE SND+1,RND(I)*32+50:POKE SND+4,17:POKE SND+4,16:NEXT I
Z 850 FOR I=1 TO 24:POKE SND+I,0:NEXT:RETURN
G 860 POKE SND+24,7:POKE SND+5,192:POKE SND+6,250
K 870 FOR I=110TO 15 STEP -1:POKE SND+1,I:POKE SND+4,17:NEXT
J 880 POKE SND+24,15:POKE SND+1,15:POKE SND+4,128:FOR I=1 TO 1500:NEXT
J 890 FOR I=1 TO 24:POKE SND+I,0:NEXT:RETURN
Z 900 IF C(16,1)>=92 THEN D=1: RETURN
G 910 IF C(7,1)<5 THEN D=2: RETURN
B 920 IF C(9,1)>69 OR C(9,1)<31 THEN D=3: RETURN
N 930 IF AGE>3000 THEN D=4: RETURN
Q 940 IF NOT (C(7,1)>77 AND C(12,1)>92 AND C(14,1)>61 AND C(16,1)<46) THEN RETURN
N 950 IF C(18,1)>92 THEN D=5: RETURN
H 960 RETURN
P 970 POKE53269,0:PRINT"{SHIFT CLR}{12 CRSRDOWN}";
M 980 IF D<5 THEN GOSUB860
A 990 IF D=5 THEN FOR I=1 TO 5:GOSUB830:NEXT
N 1000 ON D GOSUB1050,1080,1110,1130,1160
T 1010 PRINT"{3 CRSRDOWN}PLAY AGAIN (Y/N)? Y":POKE 198,0:D=0
P 1020 GET K$:IF K$="" THEN1020
V 1030 IF K$="N" OR K$="{SHIFT N}" THEN D=1:RETURN
R 1040 RETURN
J 1050 PRINT"WASTES HAVE ACCUMULATED TO A POINT WHERETHEY ARE FATAL TO THE CELL.
O 1060 PRINT"{2 CRSRDOWN}CHECK WASTE METER. USE LYSOSOMES TO";
N 1070 PRINT"INCREASE ENERGY FOR WASTE DISPOSAL.":RETURN
T 1080 PRINT"LOW EXPORTS HAVE LEAD TO INSUFFICIENT"
P 1090 PRINT"NUTRIENT SUPPLY."
K 1100 PRINT"{2 CRSRDOWN}!!WATCH THE EXPORT METER!!":RETURN
```

Continued

CELL MATES *Continued*

COMMODORE 64

```
10 PRINT"THE CELL TEMPERATURE HAS EXCE
20 PRINT"SAFETY LIMITS RESULTING
30 PRINT"IN A BREAKDOWN OF ALL FUNCTI
40 PRINT"RETURN
50 PRINTTHE CELL WAS NOT ABLE TO REPR
60 PRINTDUCT"WITHIN ITS NORMAL LIFESPAN."
70 PRINT"■CRSRDOWN■BETTER LUCK NEXT TI
80 PRINT"RETURN
90 PRINT"■IVCRSRDOWN■YOUR SCORE IS ";(30
100 INPUT"DIGIT)*10:RETURN
110 PRINT"■SHIFT CLR■";:GOSUB1460
120 FORZ=1TO24:PRINT"                    ";:NEXT:NEXT
130 FORZ=56256TO56295:POKEZ,C:NEXT
140 FORI=1TO5:POKE1131+I,32:NEXT
150 FORI=0TO18:READC(I,1),C(I,2):NE
160 FORI:FORI=0TO5:READKEYS(I):NEXT
170 NDX=136:FORI=2TO12STEP2:METR(
180 NDX=NDX:NDX=NDX+120:NEXT:METR(1)=124
190 NDX=146:FORI=3TO11STEP2:METR(
200 NDX:NDX=NDX+120:NEXT
210 FORI=1TO12:POKEMETR(I),MCHAR+9:
220 FORII=1TO14:II,MCHAR:NEXT:POKEMET
230 POKE(I)+15,MCHAR+18:NEXT
240 FORI=0TO4:POKE53248+I*2,C(I+1,1
250 POKE53249+I*2,C(I+1,2):NEXT
260 FORI=1TO12:MHST(I,1)=1:MHST(I,2)
270 =128:NEXTTO12:READZ:NDX=55296+(MET
280 FORII=1TO12:READZ:NDX=55296+(MET
290 FORII=NDXTO NDX+(Z-1):POKEII,5:N
300 EXTI
310 POKE53287,5:POKE53288,6:POKE53328
320 ,7:POKE53290,2:POKE53291,4
330 FORI:POKE53269,0:SBASE+319:READZ:POK
340 FORI:NEXTI:POKE2040+I,224+I:NEXT
350 FORII=0TO24STEP3:Z=VID+(II*40):F
360 ORII=ZTOZ+39:POKEII,67:NEXT:NEX
370 TI
380 FORI=1TO4:READZ:NDX=0:X=(I=1)*-
390 107+(I=3)*-107+(I=2)*-115+(I=4)*-11
400 5
410 FORIIIZTOZ+(24*40)STEP40:IFNDX
420 THENPOKEII+VID,X:NDX=1:GOTO13
430 FORI=1TO19:READZ:NDX=NDX+1:
440 NEXTI=1TO8:FORII=1TO19:READZ:
450 POKEII+VID+1,Z:NEXT:NEXT
460 FORI=1TO8:FORII=1TO19:READZ:
470 POKEII+VID,Z:NEXT:NEXT
480 PRINTINITIALIZING...PEEK(56334)AND254:POKE1,
490 POKE56334,PEEK(56334)AND251
500 FORI=0TO2048:POKENDX,PEEK(53248+I):NDX=NDX+1:NEXT
510 POKE334,PEEK(1)OR4:POKE56334,PEEK(56
520 )ORCBASE+8*MCHARS:FORI=1TO27:RE
530 ADZ:POKECBASE+8*MCHARS+I,Z:NEXT
540 FORII=1TO6:POKENDX+II,255:NDX=NDX+7:NEXT
550 POKEI,Z:NEXT
560 DATA1,193,193,49,49,13,13,7,7
570 DATA0,0,0,255,255,192,192,192,0,0,
0,0,0,248,248,31,31,24,248,248,0,0,
580 DATA40,60,40,60,40,100,155,100,153
590 DATA57,96,79,1,50,50,50,50,0,0,50,50,10
600 DATA35,35,20,20,0,0,15,15,50,50
610 DATA15,4,13,7,4,6,6,6,6,0,0,16,1,0,0
620 DATA7,0,1,7,0,0,16,1,
630 DATA2,69,96,6,37,64,9,21,128,24,14
640 DATA36,86,0,98,84,0,81,88,0,201,80,
650 DATA165,160,0,69,192,0,53,128,0,23
660 DATA12,0,0
670 DATA126,28,7,126,126,15,124,227,27
680 DATA193,131,27,195,3,51,198,6,54,102,1
690 DATA54,60,24,102,24,48,204,0,227,1
700 DATA1,207,12,7,28,24,28,48,48,112,96
710 DATA224,192,192,193,129,140,193,133
720 DATA195,142,63,99,24,255,62,31,255
730 DATA30,15,3,48,252,0,14,86,0,26,75,0,
740 DATA52,129,128,48,252,128,106,36,192,66,
750 DATA64,76,8,64,193,136,96,178,17,1
760 DATA196,2,32,192,128,96,76,17,64,66,
770 DATA64,97,113,192,32,10,128,55,33,
780 DATA24,140,0,12,0,122,0,7,252,0,0
790 DATA0,0,0,0,0,66,16,128,131,57,1
800 DATA72,0,0,0,192,0,0,64,4,33,8,14,11
810 DATA140,0,0,8,0,0,0,115,156,224,33,8,
820 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
830 DATA255,255,255,255,255,255,255,255,255,25
840 DATA255,224,0,7,224,0,7,224,0,7,224,0,7,22
850 DATA0,7,224,0,7,224,0,7,224,0,7,224,0,
860 DATA224,0,7,224,0,7,224,0,7,255,255,255,255,2
870 DATA255,20,255,255,255,255,0
880 DATA0,0,112,2,39,110,960,109,999,125,2
890 DATA114,980,66,113,98,66,113,380,241,20,67
900 DATA"ICORELLMATRIXES",40,"AGE:INTERNAL TEMPER
910 DATA"IMPEXTERNALSETTEMP",0,"EXPO",RT"WASTE
920 DATA"RIBOSOMEPROTEIN ENERGY",ENERGY
930 DATA"WASTEPROTEIN ENERGY PROTEIN
940 DATA112,64,64,64,64,64,110,32,32,8
950 DATA64,64,81,64,64,64,64,110,32,32
960 DATA66,81,64,64,64,64,32,109,64,32,110,1
970 DATA12,66,64,64,64,64,125,32,32,32
980 DATA26,78,119,111,111,111,182,32,32,66,
990 DATA116,109,119,80,111,111,122,80,32,32,
1000 DATA109,83,64,64,111,125,32,102,102
1010 DATA102,111,111,111,111,111,111,122,32
1020 DATA112,64,64,64,64,32,64,64,102,32,
1030 DATA102,64,64,64,64,32,32,32,32,32,32,
1040 DATA109,67,64,64,64,32,32,32,112,125
1050 DATA109,64,110,32,109,64,32,110,32,32,
1060 DATA64,125,32,64,64,64,32,32,66,32,32,1
1070 DATA109,64,64,64,64,64,64,64,125,32,32,1
1080 DATA64,109,64,64,64,64,125,32,32,32
1090 DATA64,"HOME3 CRSRRIGHT";:"QUIT (Y
1100 DATA$:IF A$="Y" THEN RETURN
1110 DATA$<>"N" THEN1930
1120 DATA"HOME3 CRSRRIGHT11 S
```

HCM

CELL MATES

IBM PC/PCjr, TANDY 1000

```
100 '**********CELL MATES**********
110 '**********CELLULAR AUTOMATA**********
120 '*********COPYRIGHT 1985*********
130 '*********ALLEY PUBLISHING CO.*********
140 '*********BY WILLIAM K. BALTHROP*********
150 '*********AND THE HOME COMPUTER MAGAZINE*********
160 '*********VERSION 2.1 for PCjr with BASIC adapter*********
170 '*********DOS 2.1 and either BASIC or*********
180 '*********IBM IBM PCjr with CARTRIDGE BASIC*********
190 '*********COLOR/GRAPHICS ADAPTER and*********
200 '*********COLOR MONITOR or*********
210 '*********TANDY 1000 with GW BASIC*********
220 '*********COLOR 15,1:WIDTH 40:CLS:LOCATE ,,0:KEY OFF:SCREEN 2,1:PRINT"
230 '*********CONTINUE":CLOCATE 14,2:PRINT"PRESS ANY KEY":CHR$(217);
240 '
250 '
260 '
270 '
280 '
290 '
300 '
```

Continued

TYPE-IN LISTINGS

CELL MATES *Continued*



Continued

CELL MATES *Continued*

IBM PC/PCjr, TANDY 1000

```
A 1260    Y=115:X=RIBO.E:GOSUB 1410:RIBO.E.OL
Y 1270    Y=RIB9:X=EX.E:GOSUB 1410:EX.E.OLD=EX
N 1280    YOLD=163:X=WASTE.E:GOSUB 1410:WASTE.E.
X 1290    YPRO.E=187:X=PRO.E:GOSUB 1410:PRO.E.OLD=
A 1300    YNUT.L=43:X=NUT.L:GOSUB 1420:NUT.L.OLD=N
W 1310    YIPRO=67:X=IPRO:GOSUB 1420:IPRO.OLD=IPR
X 1320    YEX.TEMP=91:X=EX.TEMP:GOSUB 1420:EX.TEMP.O
S 1330    YRIBO.L=115:X=RIBO.L:GOSUB 1420:RIBO.L.OL
M 1340    YEX.L=139:X=EX.L:GOSUB 1420:EX.L.OLD=EX
H 1350    YWASTE.L=163:X=WASTE.L:GOSUB 1420:WASTE.L.
N 1360    YPRO.L=187:X=PRO.L:GOSUB 1420:PRO.L.OLD=
A 1370    YMITO.Y=7:MITO.Y=12:AGE=0:ATP=0:MF=1
E 1380    YCIS=1:I$="Mn Gg"(1,2):C(129,4,2)=C(3,1,2):C(5,1)=71:C(5,2
C 1390    YDRT=0:RETURN:'RETURN
P 1400    YX*1.5+16,Y):RETURN
D 1410    YX*11,RENUM+2,0:R4G4H4R4BDP2,2"
X 1420    YSCREEN 2,0:DRAW "BM100,100C3BR16R2FD3GD2GD2E"
B 1430    YINITIALIZE:DIM(26):FOR I=1 TO 26:READ A$(I):NEXT I:RETURN
A 1440    YC4,0):DRAW "BM119,119;XA$;":GET(2,0)-(10,5
I 1450    YDRAW "BM119,119;XA$;":GET(2,0)-(10,5
X 1460    YDRAW "BM100,100C3BR16R2FD3GD2GD2E"
S 1470    YLINE (156,71),,B
X 1480    YLINE (156,71),,B
M 1490    YCAT:E 24:PRINT"PRESS ENTER TO CONTINUE"
P 1500    YD$="R":D$=D$+D$+D$+D$+D$+"M+3,+0NU6
E 1510    YFOR D=1 TO 8:READ X1,Y1,X2,Y2:LIN
L 1520    YEXT D:FOR I=1 TO 24:PRINT:NEXT:LINE (0
K 1530    YLINE (161,28)-(
W 1540    YDRAW "BM
A 1550    YDRAW "BM
A 1560    YDRAW "BM
L 1570    YDRAW "BM
M 1580    YDRAW "BM
Y 1590    YDRAW "BM
O 1600    YPLAY"O
A 1610    YPUT(
I 1620    YDRAW "BM11,44C3G2D2G2D2G2D6F2D5F2D2F2D2F2R4
M 1630    YDRAW "L4":PAINT (11,49),CHR$(230)+C
M 1640    YLINE (127,51)-(151,68),0,BF:CIRCLE (
M 1650    YPAINT (74,35),CHR$(138)+CH
```

```
G 1660    YDRAW "BM119,13C2EDRBRBD2FURBR3BU2EDDG
          YRBF2RBLBU2HDLBG2GULBH2HDLBG2GULBH2HDLBG2GULBH2HDLBG2GULBH2HD
          YULPAINT (2,5),CHR$(204)+CHR$(51)
D 1670    YRESTORE:DRAW B$+D$+D$+D$+D$:DRAW
L 1680    YRETURN (X,Y):DRAW B$+D$+D$+D$+D$:RETURN
I 1690    YDATA 51,90,107,98,281,42,315,50,281
I 1700    YDATA 315,74,203,90,278,98,303,114,31
          YDATA 56,138,315,146,165,162,234,17
A 1710    YDATA 186,315,194
A 1720    YDATA "RIB. ENERGY"
F 1730    YDATA "WASTE ENERGY"
Z 1740    YDATA "NUT TEMP. "
J 1750    YDATA "IPRO. "
Y 1760    YDATA 80,217601,80
          YDATA 217601,84
G 1770    YDATA 165501,166005,308487,165501,308487,165501,308487,165501,
T 1780    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
C 1790    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
T 1800    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
N 1810    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
A 1820    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
B 1830    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
P 1840    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
C 1850    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
Q 1860    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
M 1870    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
J 1880    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
N 1890    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
W 1900    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
          YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
J 1910    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
Z 1920    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
M 1930    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
W 1940    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
G 1950    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
Y 1960    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
A 1970    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
D 1980    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
Y 1990    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
Q 2000    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
Y 2010    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
N 2020    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
W 2030    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
I 2040    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
F 2050    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
N 2060    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
Y 2070    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
X 2080    YDATA 308487,165501,308487,165501,308487,165501,308487,165501,
```

HCM

CELL MATES

TI-99/4A

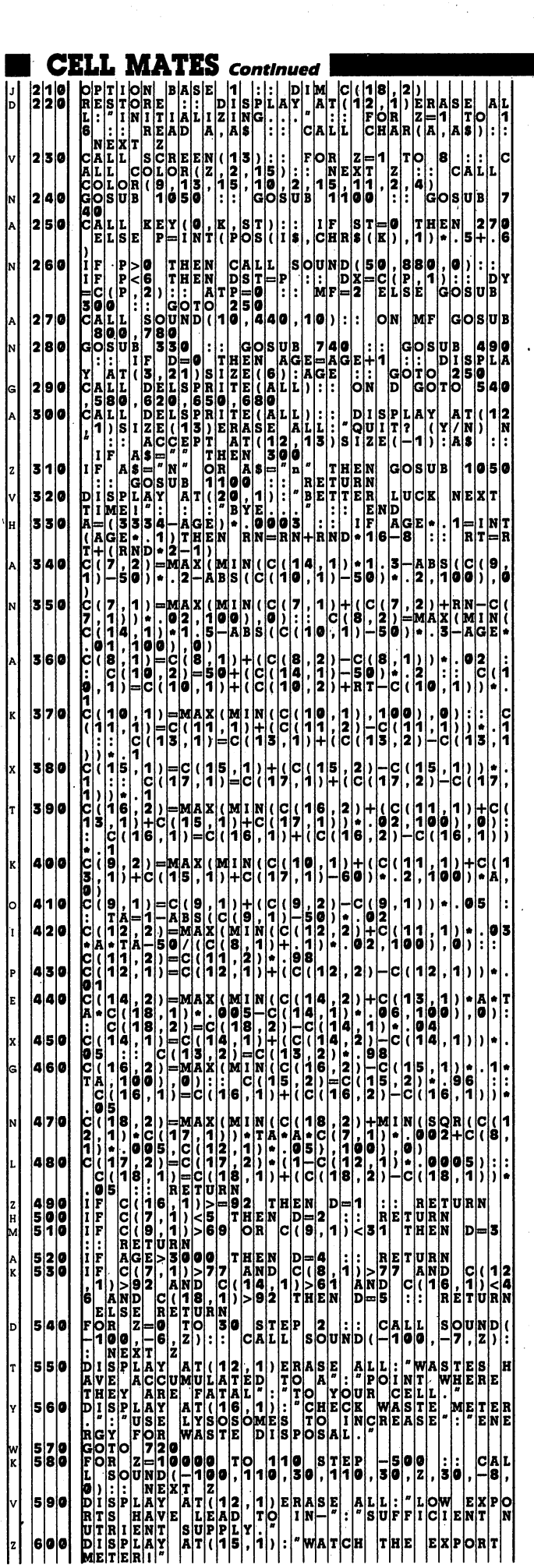
```
O 100     !  **********
X 110     !  *  CELL MATES  *
E 120     !  **********
J 130     !  COPYRIGHT 1985
M 140     !  EMERALD VALLEY PUBLISHING CO.
N 150     !  BY WILLIAM K. BALTHROP
M 160     !  HOME COMPUTER MAGAZINE

J 170     !  VERSION 5.6.1
E 180     !  TI EXTENDED BASIC
J 190     !
Q 200     RANDOMIZE :: DISPLAY AT(12,5)ERASE ALL:"PRESS ENTER TO CONTINUE"
          ::CALL KEY(3,K,S):: IF K<>13 THEN 200
```

Continued

TYPE-IN LISTINGS

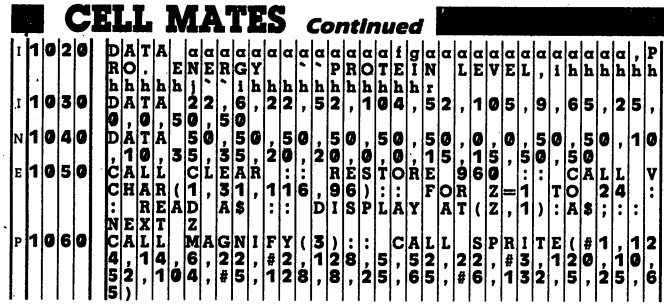
TYPE-IN LISTINGS



Continued

CELL MATES *Continued*

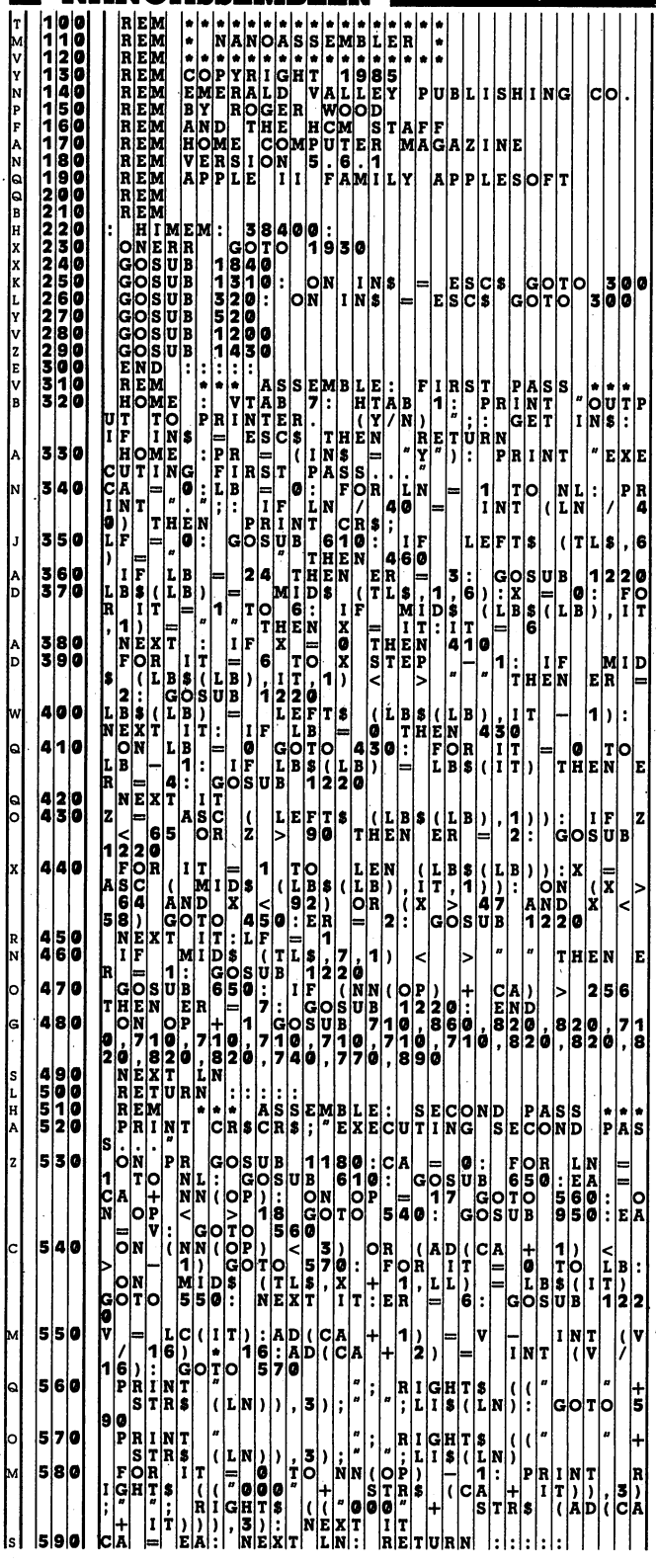
TI-99/4A



HCM

NANOASSEMBLER

APPLE II Family

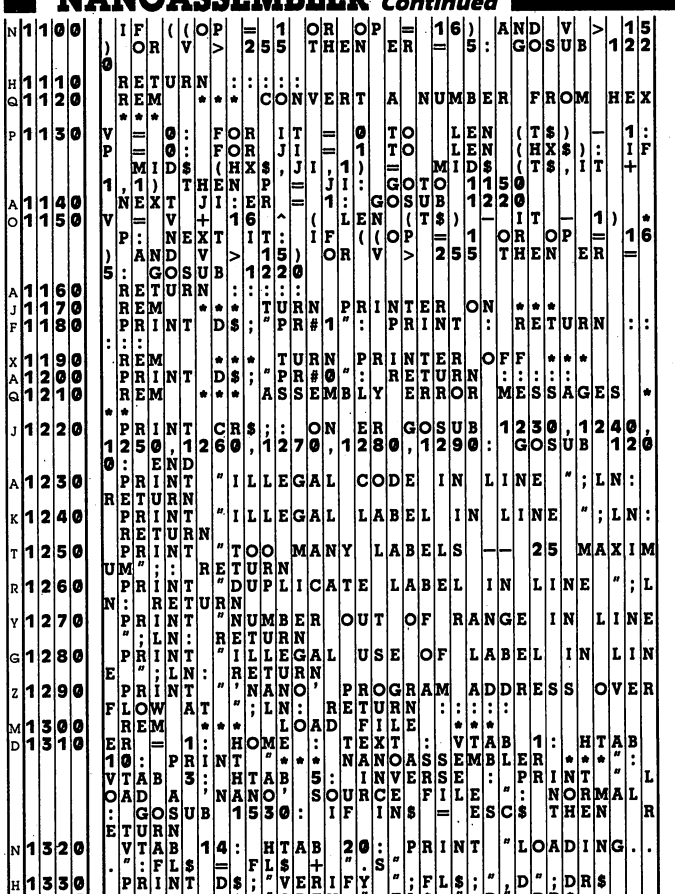


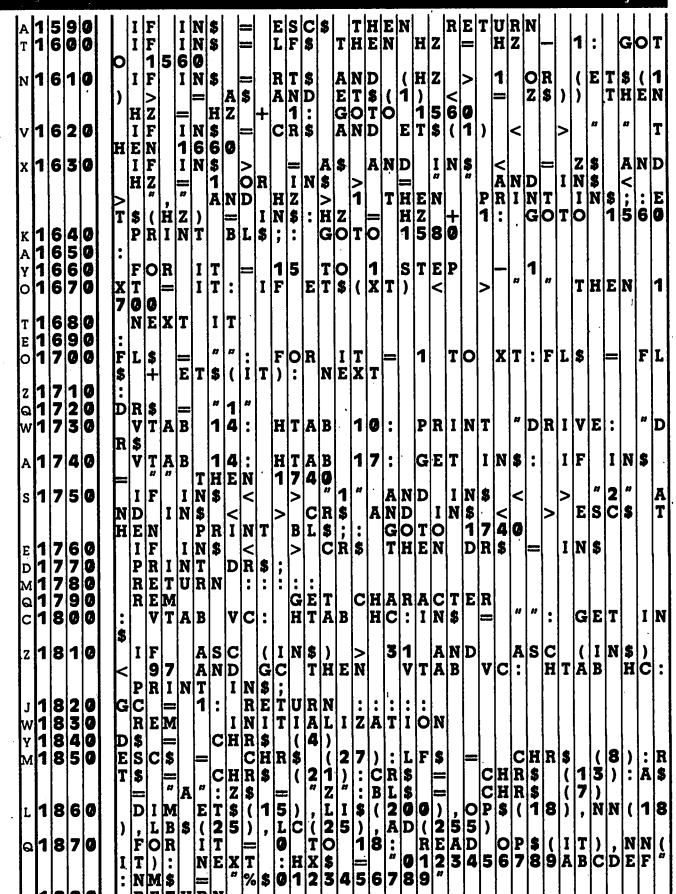
Continued

TYPE-IN LISTINGS

NANOASSEMBLER *Continued*

APPLE II Family

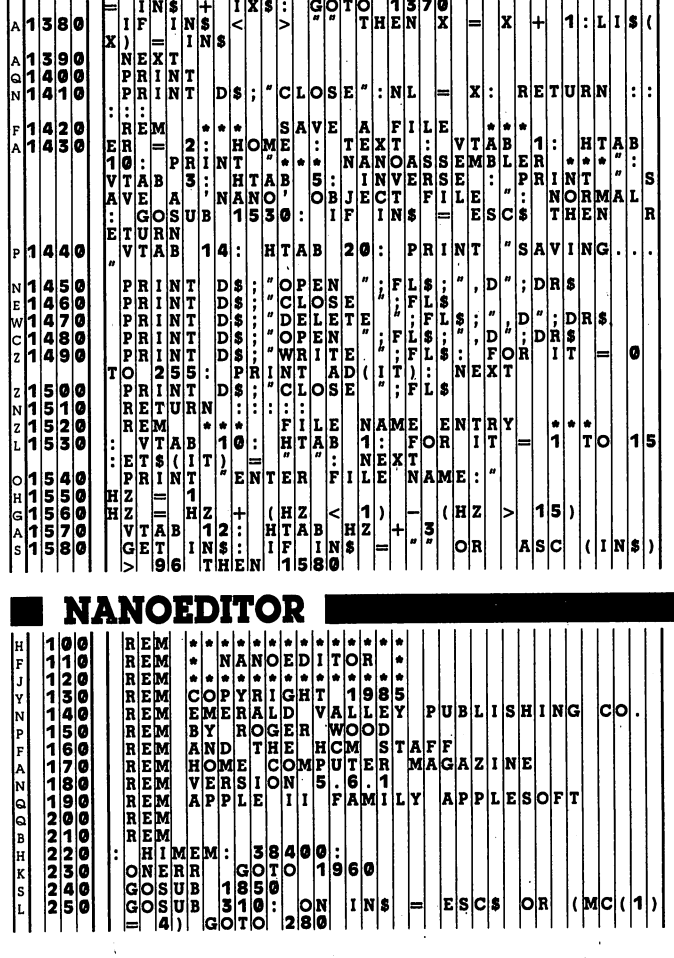


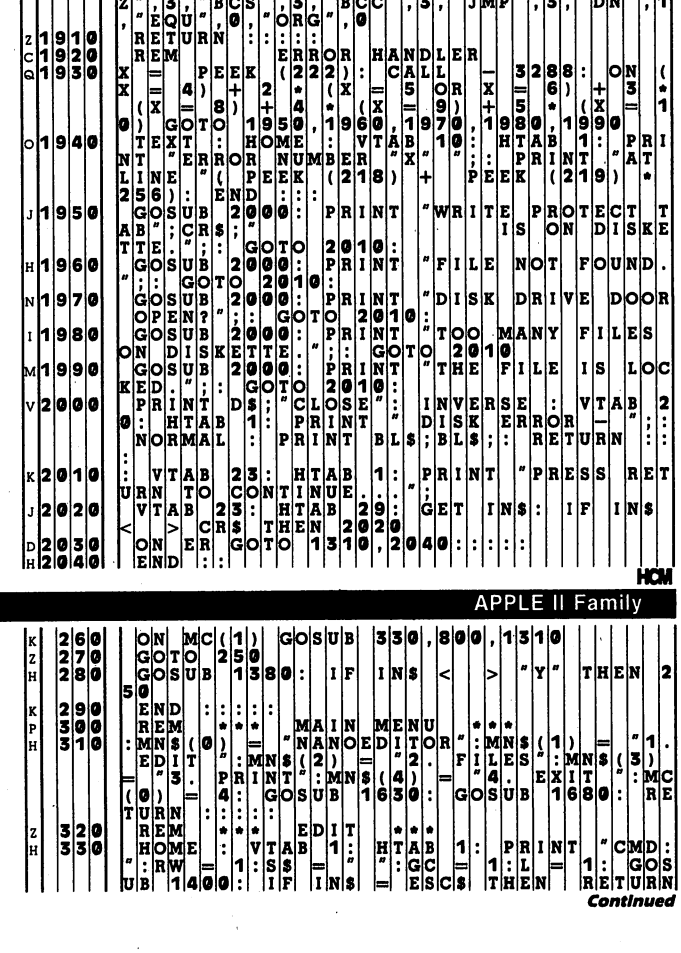


HCM

NANOEDITOR

APPLE II Family





Continued



Continued

NANOEDITOR *Continued*

APPLE II Family

```
P 1490  S$ = "": FOR EI = 1 TO P: S$ = S$ + "
  ET$(EI): NEXT: GOTO 1410
A 1500  HZ = HZ - 1: GOTO 1410
N 1510  HZ = HZ + 8: GOTO 1410
P 1520  ON > L0 GOTO 1410: GOTO HZ
  ON 34 GOTO 1410: GOTO HZ
J 1530  ON L0 GOTO 1410: GOTO HZ
  IF RW: HTAB HZ(HZ): PRINT ET$(HZ);
T 1540  BK$: VTAB HZ(HZ): PRINT ET$(HZ);
  TO L0 THEN ET$(HZ) = BK$:
  OL0 = HZ: L0 = HZ: (L0 < L): FOR EI = L0 TO 1(
  TO EI): $ET$(EI + 1): HZ = HZ - 1: NEXT EI
  ET: ET$(EI + 1) = BK$: VTAB HZ: PRINT ET$(HZ);
  ET: VTAB (EI + 1): HTAB HZ(HZ): PRINT ET$(H
  Z): GOTO 1410
H 1560  IF RW: HTAB HZ(HZ): PRINT ET$(HZ): BK$: VTAG
  BO TO OR 1410: HZ TO L0 - 1: ET$(EI): ET
  $(EI + 1): VTAB HZ: PRINT BK$: VTAG
L 1570  $(EI + 1): VTAB HZ: PRINT ET$(EI): ET
  INT ET$(EI): NEXT: ET$(L0): BK$(L0)
  V: HTAB HZ(L0): PRINT ET$(L0)
  : GOTO 1410
M 1580  HZ = HZ + 1: GOTO BK$: L0 = L: CALL EO: ON
  HZ > 33 GOTO 1590: VTAB RW + 1: HTA
  B 7: CALL EOL: GOTO 1410
Q 1590  XT = HZ: GOTO 1410
G 1600  HZ = 1: GOTO 1410
A 1610  RETURN
Y 1620  REM ** CONFIGURE MENU **
P 1630  :X = 11 - MC(0): IF MC(0) = 0 THEN
  1650
K 1640  FOR IT = 1 TO MC(0): MN(IT, 0): NEXT
  2 + LEN (MN$(IT)): MN$(IT, 2): NEXT
  MN$(0)
N 1650  RETURN
T 1660  REM ** OPERATE MENU **
F 1670  REM
A 1680  LV = 0: HOME: VTAB 1: HTAB 1: INT (
  39) - LEN (MN$(0))): PRINT MN$
  (0): VTAB 24: HTAB 9: PRINT "PRESSE
  V: VTAB 22: HTAB 1: CALL EOL: RETUR
  NN
R 1690  MC(1) = 1: MC(2) = MC(1): FOR IT = 1
  TO MC(0): VTAB MN(IT, 0): HTAB MN(I
  T, 1): PRINT MN$(IT);: NEXT
C 1700  : VTAB MN(MC(2), 0): HTAB MN(MC(2), 1)
  : TAB MN(MC(1), 0): HTAB MN(MC(1), 1):
  INVERSE : PRINT MN$(MC(2)): MC(2) =
  MC(1): MC(1) = MC(0): NORMAL : PRINT
  MN$(MC(1)): RETURN
F 1710  VC = 0: ES = GOSUB 1790: IF RETURN IN$ =
  CR$ OR IN$ = "" THEN RETURN
F 1720  IN$ = LF$ OR IN$ = UP$ THEN MC(1) = MC(
  1) + 1: MC(0) = MC(1): GOTO 1700
Z 1730  IF IN$ = RT$ OR IN$ = "" THEN MC(1) = MC(
  1) - 1: MC(0) = MC(1): GOTO 1700
X 1740  FOR MI = 1 TO MC(0): IF IN$ = MID$ (MN$(MI), 1, 1) THEN LV = MI: MC(1) =
```

```
Q 1750  NEXT MI
C 1760  GOTO 1710
L 1770  RETURN
J 1780  REM ** GET TAB VC: HTAB VC: GET IN
E 1790  $: IF IN$ = "" THEN 1790: IF ASC (IN$) <
  97 AND ASC (IN$) < 123 THEN IN$ = CHR$ (
  ASC (IN$) - 32): GOTO 1790: RETURN
K 1800  REM ** SCAN INPUT **
A 1810  PR = 1: PRINT "": INVERSE : PRINT "
N 1820  REM ** CURSOR CONTROL **
G 1830  $: IF IN$ = CHR$ (8) THEN ES = "A": RETU
  RN : REM ** ESPON **
  $: IF IN$ = CHR$ (21) THEN ES = "Z": RETU
  RN : REM ** ESPON **
  $: IF IN$ = CHR$ (11) THEN ES = "U": RETU
  RN : REM ** ESPON **
  $: IF IN$ = CHR$ (10) THEN ES = "I": RETU
  RN : REM ** ESPON **
X 1840  REM ** ESPON **
F 1850  CHR$ (4): "
D 1860  CHR$ (27): LF$ = CHR$ (10): BL$ = CHR$
  (9): RT$ = CHR$ (21): "Z" = CHR$ (26): "
  U" = CHR$ (85): "P" = CHR$ (80): "
L 1870  CHR$ + LF$ + RT$ + "4" + X$: "
  AD (X$): "
U 1880  (66), MN$(8), MN(8, 1), MC(2), H
N 1890  = 7 + IT: H
L 1900  = 1
  FOR IT = 1 TO MN(0): HTAB MN(IT, 0): PRINT
  MN$(IT, 1): NEXT
L 1910  REM ** DATA **
Q 1920  REM "NAME", "ADDRESS", "PHONE"
Z 1930  DATA "", "", ""
G 1940  REM ** RETURN **
J 1950  REM ** HANDLER **
A 1960  = 4): 8): + 2: 6): + 3: 8): + 1
  = 0: 0): 1980: 2000: 2010: 2020: 2030
  : 2040: 2050: 2060
```

```
Q 1750  NEXT MI: RETURN
C 1760  GOTO 1710: RETURN
L 1770  REM ** GET TAB VC: HTAB VC: GET IN$: IF
J 1780  $ < "a" OR IN$ > "z" THEN RETURN
E 1790  IN$ = CHR$ (ASC (IN$) - 32): RETURN
K 1800  REM ** SCAN INPUT **
A 1810  PR = 1: PRINT "": INVERSE : PRINT "
N 1820  REM ** CURSOR CONTROL **
G 1830  $: IF IN$ = CHR$ (8) THEN ES = "A": RETU
  RN : REM ** ESPON **
  $: IF IN$ = CHR$ (21) THEN ES = "Z": RETU
  RN : REM ** ESPON **
  $: IF IN$ = CHR$ (11) THEN ES = "U": RETU
  RN : REM ** ESPON **
  $: IF IN$ = CHR$ (10) THEN ES = "I": RETU
  RN : REM ** ESPON **
X 1840  REM ** ESPON **
F 1850  CHR$ (4): "
D 1860  CHR$ (27): LF$ = CHR$ (10): BL$ = CHR$
  (9): RT$ = CHR$ (21): "Z" = CHR$ (26): "
  U" = CHR$ (85): "P" = CHR$ (80): "
L 1870  CHR$ + LF$ + RT$ + "4" + X$: "
  AD (X$): "
U 1880  (66), MN$(8), MN(8, 1), MC(2), H
N 1890  = 7 + IT: H
L 1900  = 1
  FOR IT = 1 TO MN(0): HTAB MN(IT, 0): PRINT
  MN$(IT, 1): NEXT
L 1910  REM ** DATA **
Q 1920  REM "NAME", "ADDRESS", "PHONE"
Z 1930  DATA "", "", ""
G 1940  REM ** RETURN **
J 1950  REM ** HANDLER **
A 1960  = 4): 8): + 2: 6): + 3: 8): + 1
  = 0: 0): 1980: 2000: 2010: 2020: 2030
  : 2040: 2050: 2060
```

HCM

NANOASSEMBLER

ATARI 800/800XL/130XE

```
N 100  REM ** ASSEMBLER **
N 110  REM ** NANOASSEMBLER **
V 120  REM **********
M 130  REM COPYRIGHT 1985
X 140  REM COMPUTER ALLEY PUBLISHING CO
Z 150  REM BY ROGER WOOD
L 160  REM AND THE HCM STAFF
K 170  REM HOME COMPUTER MAGAZINE
F 180  REM VERSION 5.6.1
M 190  REM ATARI BASIC FOR THE 800, 800XL,
  130XE
I 200  REM
E 210  GRAPHICS 0: SETCOLOR 2, 0, 0: TRAP 2140
  :CLOSE #5: OPEN #5, 4, 0, "K:": POKE 752
  ,1
U 220  POSITION 7, 0: PRINT "◤The Nanoassemb
  ler◤": POSITION 8, 5: PRINT "Copyright
  1985"
N 230  DIM OP$(79), NN(19), LINE$(1500), LB$
  (17,75), AD(255), A$(76), DEV$(20), FL$(
  9), TVL$(9), OS$(10), OT$(10), OU
T 240  DIM EXT$(4)
N 250  GOSUB 1980: RESTORE 1880: GOSUB 1860
H 260  GOSUB 1950
E 270  IF KEY$>50 THEN 260
P 280  IF KEY$="49" THEN LF=1: DEV$(1,2)="D:"
  EXT$="S": GOTO 2050: GOTO 300
E 290  DEVICE$="D:": ? "PLAY ON": PLAY ON
  RECORD #3: ? "PRESS RETURN": DEV$
Y 300  CLOSE #3: OPEN #3, 4, 0, DEV$
Y 310  GET #3, NL: FOR I=1 TO NL: INPUT #3; A$
  :LINE$(I*76,(I*76)+75)=A$: NEXT I
D 320  CLOSE #3: GOSUB 1180
J 330  GOSUB 4300: REM ASSEMBLE
C 340  RESTORE 1900: GOSUB 1860
B 350  GOSUB 1950: IF KEY$="27" THEN 1240
```

```
E 360  IF KEY$="49" THEN 350
S 370  IF KEY$="49" THEN LF=1,2)="D:": "
  EXT$="S": GOTO 390
A 380  DEVICE$="D:": ? "PRESS PLAY/RECO
  RD": ? "PRESS RETURN"
L 390  CLOSE #3: OPEN #3, 8, 0, DEV$
M 400  CLOSE #3, AD(I): NEXT I
H 410  CLOSE : GOTO 120
K 420  REM ** ASSEMBLE **
H 430  PRINT "INTERACTIVE FIRST PASS"
B 440  CA=0: LB=1
F 450  FOR LN=1 TO NL
L 460  FOR LAG=0: GOSUB "
F 470  IF LB$(1,6)="      " THEN 650
F 480  IF LB$(LB*7)="9600" THEN 960
G 490  IF LB$(LB*7)="5" OR LB$(LB*7)="6" TO
  5: IF LB$(1,6)="      " THEN LB$(LB*7)
  =IT: GOTO 520
L 500  IF LB$(LB*7)="SEM" THEN LABEL(LB)=
  LB*7: LB$(LB*7)="      "
Y 510  NEXT LB
C 520  IF LABEL(LB)=6 THEN 550
A 530  FOR I=LB*7 TO LB*7+IT: IF I<>" " THEN
  LB$(I)=" ": NEXT I
T 540  IT=0
N 550  IF LB$="" THEN 580
Y 560  FOR I=LB*7 TO LB*7+LABEL(LB)-1: IF
  I<>" " THEN IT=IT+1: NEXT I
W 570  IF ASC(LB$(1,1))<65 OR ASC(LB$(1,1)
A 580  )>90 THEN 950: LABEL(LB)=LINE: LB
  =LB+1: GOTO 970
M 590  FOR I=LB TO LB+IT: LB$(I)=" ": NEXT
  I
P 600  IF (X>64 AND X<91) OR (X>47 AND X<5
```

Continued

```
U  610 GOTO 950
U  620 NEXT IT
T  630 LB$((LB*7)+6,(LB*7)+6)=CHR$(CA)
M  640 LFLAG=1
N  650 IF TL$(7,7)<>" " THEN 940
I  660 GOSUB 1120
Y  670 IF CA+NN(OP)>256 THEN 1000
Q  680 ON OP GOTO 1250,1430,1370,1370,1250,1250,1250,1250,1250,1250,1250,1370,1370,1370,1370,1370,1280,1320,1460
A  690 IF LFLAG=0 THEN 710
E  700 LB=LB+1
X  710 CA=CA+NN(OP):NEXT LN
B  720 ? "EXECUTING SECOND PASS"
V  730 CA=0
T  740 FOR LN=1 TO NL
Y  750 GOSUB 1070:GOSUB 1120:EA=CA+NN(OP)
L  760 IF OP=18 THEN 890
G  770 IF OP<>19 THEN 810
R  780 GOSUB 1500
J  790 EA=V
G  800 GOTO 890
T  810 IF NN(OP)<3 THEN 900
J  820 IF AD(CA+1)<>-1 THEN 900
K  830 FOR IT=1 TO LB:IF TL$(X+2,LL)=LB$(IT*7,(IT*7)+LABLL(IT)-1) THEN 850
N  840 NEXT IT:GOTO 990
V  850 V=ASC(LB$((IT*7)+6,(IT*7)+6))
B  860 AD(CA+1)=V-INT(V/16)*16
H  870 AD(CA+2)=INT(V/16)
M  880 GOTO 900
A  890 PRINT #PR;"       ";LN;"   ";LINE$(LN*76,(LN*76)+75):GOTO 920
W  900 PRINT #PR;"       ";LN;"   ";LINE$(LN*76,(LN*76)+75)
B  910 FOR IT=0 TO NN(OP)-1:GOSUB 1020:PRINT #PR;OT$:NEXT IT
B  920 CA=EA:NEXT LN
L  930 PRINT "PRESS ANY KEY TO CONTINUE":GOSUB 1950:CLOSE #PR:RETURN
U  940 PRINT #PR;"ILLEGAL CODE IN LINE ";LN:GOTO 1010
Q  950 PRINT #PR;"ILLEGAL LABEL IN LINE ";LN:GOTO 1010
I  960 PRINT #PR;"TOO MANY LABELS-- 25 MAXIMUM":GOTO 1010
F  970 PRINT #PR;"DUPLICATE LABEL IN LINE ";LN:GOTO 1010
A  980 PRINT #PR;"NUMBER OUT OF RANGE IN LINE ";LN:GOTO 1010
A  990 PRINT #PR;"ILLEGAL USE OF LABEL IN LINE ";LN:GOTO 1010
N 1000 PRINT "ADDRESS OUT OF RANGE IN LINE ";LN:GOTO 1010
Y 1010 GOSUB 1960:GOTO 1240
T 1020 OT$=STR$(CA+IT):S$=OT$:OT$="000":OT$(4-LEN(S$),LEN(OT$))=S$
D 1030 OU$=STR$(AD(CA+IT)):S$=OU$:OU$="000":OU$(4-LEN(S$),LEN(OU$))=S$
R 1040 OT$(LEN(OT$)+1)=" "
I 1050 OT$(LEN(OT$)+1)=OU$
F 1060 RETURN
N 1070 TL$=LINE$((LN*76),(LN*76)+20)
I 1080 FOR IT=LEN(TL$) TO 1 STEP -1
K 1090 IF TL$(IT,IT)<>" " THEN 1110
F 1100 TL$=TL$(1,LEN(TL$)-1):NEXT IT
R 1110 LL=IT:RETURN
O 1120 FOR IT=1 TO 19:IF 8+(OP(IT)-1)>LL THEN 1140
T 1130 IF TL$(8,8+(OP(IT)-1))=OP$(4*IT,(4*IT)+(OP(IT)-1)) THEN 1150
G 1140 NEXT IT:GOTO 940
O 1150 OP=IT:X=8+(OP(IT)-1)
Q 1160 IF OP=10 THEN IF TL$(X+1,X+1)="G" THEN NEXT IT
X 1170 RETURN
L 1180 PRINT "send assembly to:":PRINT "1. screen":PRINT "2. printer"
J 1190 GOSUB 1920
L 1200 IF KEY=50 THEN 1220
X 1210 PR=6:RETURN
D 1220 POKE 764,255:PRINT "turn on printer and <press a key>":CLOSE #7:OPEN #7,8,0,"P:":PR=7:GOSUB 1950:RETURN
F 1230 REM END PROGRAM
U 1240 PRINT :GRAPHICS 0:TRAP 2340:END
B 1250 IF X+1<>11 THEN 940
Z 1260 AD(CA)=OP-1
U 1270 GOTO 690
L 1280 GOSUB 1500
B 1290 IF V=-1 THEN 990
P 1300 AD(CA)=V
B 1310 GOTO 690
O 1320 GOSUB 1500
U 1330 IF V=-1 THEN 990
M 1340 IF LFLAG=0 THEN 940
J 1350 LB$((LB*7)+6,(LB*7)+6)=CHR$(V)
A 1360 GOTO 690
D 1370 GOSUB 1500:AD(CA)=OP-1
R 1380 IF V<>-1 THEN 1400
Q 1390 AD(CA+1)=-1:AD(CA+2)=-1:GOTO 690
I 1400 AD(CA+1)=V-INT(V/16)*16
G 1410 AD(CA+2)=INT(V/16)
R 1420 GOTO 690
M 1430 GOSUB 1500
S 1440 IF V=-1 THEN 990
L 1450 AD(CA)=OP-1:AD(CA+1)=V:GOTO 690
V 1460 GOSUB 1500
A 1470 IF V=-1 AND LFLAG=1 THEN 990
U 1480 CA=V
C 1490 GOTO 690
E 1500 IF TL$(X+1,X+1)<>" " THEN 980
A 1510 IF TL$(X+2,X+2)=" " THEN 940
G 1520 IF TL$(X+2,X+2)="%" THEN 1570
H 1530 IF TL$(X+2,X+2)="$" THEN 1650
W 1540 IF TL$(X+2,X+2)>"/" AND TL$(X+2,X+2)<":" THEN 1780
S 1550 V=-1
U 1560 RETURN
W 1570 REM BINARY
V 1580 VLU$(1,LL-(X+2))=TL$(X+3,LL):V=0
D 1590 FOR I=X+3 TO LL-1
C 1600 SQUR=1:FOR POWR=1 TO LL-I:SQUR=SQUR*2:NEXT POWR
V 1610 V=V+VAL(VLU$(I-(X+2),I-(X+2)))*SQUR
J 1620 NEXT I
G 1630 V=V+VAL(VLU$(LL-(X+2),LL-(X+2)))
L 1640 RETURN
N 1650 REM HEX
N 1660 VLU$="":VLU$(1,LL-(X+2))=TL$(X+3,LL)
C 1670 IF VLU$(1,1)=" " THEN 980
N 1680 FOR I=1 TO LL-(X+2)
Y 1690 IF VLU$(I,I)>"@" OR VLU$(I,I)<"G" THEN 1720
S 1700 IF VLU$(I,I)>"/" OR VLU$(I,I)<":" THEN 1720
X 1710 GOTO 980
E 1720 NEXT I:IF LEN(VU$)=1 THEN VU$(1,2)=VU$(1,1):VU$(1,1)="0"
B 1730 IF VLU$(1,1)>"@" AND VLU$(1,1)<"G" THEN V=16*(ASC(VLU$(1,1))-55):GOTO 1750
C 1740 V=16*VAL(VLU$(1,1))
I 1750 IF VLU$(2,2)>"@" AND VLU$(1,1)<"G" THEN V=V+ASC(VLU$(2,2))-55:GOTO 1770
Y 1760 V=V+VAL(VLU$(2,2))
H 1770 RETURN
Y 1780 REM DEC
D 1790 VLU$(1,LL-(X+1))=TL$(X+2,LL)
C 1800 FOR I=1 TO LL-(X+1)
D 1810 IF VLU$(I,I)<"0" OR VLU$(I,I)>"9" THEN 980
B 1820 NEXT I
G 1830 V=VAL(VLU$(1,LL-(X+1)))
J 1840 RETURN
W 1850 REM PRINT MENU
S 1860 PRINT "▨ESC CTRL <▨":READ NI
Y 1870 FOR IT=1 TO NI:READ A,A$:POSITION A,2*IT:PRINT A$;:NEXT IT:RETURN
B 1880 DATA 7,5,▨The NanoAssembler▨,1,,1,load source file from:
G 1890 DATA 8,(1) Disk (D1:),8,(2) Cassette (C:),1,,5,your choice:
Y 1900 DATA 7,5,▨The NanoAssembler▨,1,,7,save object file to:
F 1910 DATA 8,(1) Disk (D1:),8,(2) Cassette (C:),1,,5,your choice:
F 1920 GOSUB 1950
A 1930 IF KEY=49 OR KEY=50 THEN RETURN
X 1940 GOTO 1920
A 1950 GET #5,KEY:RETURN
I 1960 FOR DELAY=1 TO 500:NEXT DELAY:RETURN
Z 1970 REM PROGRAM INIT
P 1980 RESTORE 2030:FOR IT=1 TO 19:READ A$,NN,OP:OP$(IT*4,(IT*4)+3)=A$:NN(IT)=NN:OP(IT)=OP:NEXT IT
L 1990 FOR IT=0 TO 255:AD(IT)=0:NEXT IT:VLU$="00000000"
J 2000 LINE$(1)=" ":LINE$(15200)=" ":LINE$(2)=LINE$:LB$(1)=" ":LB$(175)=" ":LB$(2)=LB$
J 2010 TL$(1)=" ":TL$(76)=" ":TL$(2)=TL$
O 2020 RETURN
Z 2030 DATA ADD,1,3,LDA#,2,4,LDA,3,3,STA,3,3,TAB,1,3,TBA,1,3,RRC,1,3,RLC,1,3,AND,1,3,OR,1,2,XOR,1,3
O 2040 DATA BZ,3,2,BNZ,3,3,BCS,3,3,BCC,3,3,JMP,3,3,DN,1,2,EQU,0,3,ORG,0,3
M 2050 REM GET FILENAME
O 2060 TRAP 2060:PRINT :PRINT "Input filename";:INPUT FL$:IF LEN(FL$)>8 THEN FL$(9,LEN(FL$))=""
O 2070 IF LEN(FL$)<1 THEN 2060
F 2080 IF LEN(FL$)<2 THEN 2100
C 2090 IF FL$(1,2)="D:" THEN 2110
C 2100 DEV$=FL$:FL$(1,2)="D:":FL$(3,2+LEN(DEV$))=DEV$
Y 2110 IF EXT$="" THEN DEV$=FL$:RETURN
W 2120 DEV$=FL$:DEV$(LEN(DEV$)+1)=EXT$
S 2130 TRAP 2140:RETURN
P 2140 REM ERROR
N 2150 X=PEEK(195):Y=PEEK(187)*256+PEEK(186):POSITION 1,22:IF X=144 THEN PRINT "▨DISK I/O ERROR▨":GOTO 2310
F 2160 IF X<160 OR X>171 THEN 2300
N 2170 ON (X-159) GOTO 2180,2190,2200,2210,2220,2230,2240,2250,2260,2270,2280,2290
Q 2180 PRINT "▨DRIVE NUMBER UNKNOWN▨":GOTO 2310
Y 2190 PRINT "▨TOO MANY FILES OPEN▨":GOTO 2310
Q 2200 PRINT "▨DISK FULL▨":GOTO 2310
V 2210 PRINT "▨UNRECOVERABLE I/O ERROR▨":GOTO 2310
```

Continued

TYPE-IN LISTINGS

NANOASSEMBLER *Continued*

ATARI 800/800XL/130XE

```
V 2220  PRINT "◢FILE NUMBER MISMATCH◢":GOTO
        2310
L 2230  PRINT "◢BAD FILE NAME◢":GOTO 2310
O 2240  PRINT "◢POINT DATA LENGTH ERROR◢":G
        OTO 2310
Z 2250  PRINT "◢FILE LOCKED◢":GOTO 2310
T 2260  PRINT "◢UNKNOWN XIO COMMAND◢":GOTO
        2310

L 2270  PRINT "◢DIRECTORY FULL◢":GOTO 2310
F 2280  PRINT "◢FILE NOT FOUND◢":GOTO 2310
I 2290  PRINT "◢POINT INVALID◢":GOTO 2310
R 2300  PRINT "◢ERROR NUMBER◢";X;"◢OCCURR
        ED◢"
M 2310  GOSUB 1960
N 2320  POP :TRAP 2140:IF LF=1 THEN 250
A 2330  GOTO 340
```

HCM

NANOEDITOR

ATARI 800/800XL/130XE

```
TDNMXZLKFM 100  REM ***********
           110  REM *  NANOEDITOR  *
           120  REM ***********
           130  REM COPYRIGHT 1985
           140  REM CEMERALD VALLEY PUBLISHING CO
           150  REM BY ROGER WOOD
           160  REM AND THE HCM STAFF
           170  REM HOME COMPUTER MAGAZINE
           180  REM VERSION 5.6.1
           190  REM ATARI BASIC FOR THE 800, 800XL,
                130XE
           200  GRAPHICS 0:POKE 82,0:TRAP 1780
           210  DIM LINE$(152000),C$(15),A$(76),CMD$
                (5),CMD(5),LN$(4),S$(20),FL$(20),DE
                V$(20),KEY$(3)
           220  LINE$(1)="":LINE$(152000)="":LINE$
                (2)=LINE$
           230  CLOSE #4:OPEN #4,4,0,"S:":CLOSE #5:
                OPEN #5,4,0,"K:":SETCOLOR 2,0,0:GOS
                UB 1640
           240  PRINT "◢ESC CTRL <■":POSITION 13,0:
                PRINT "◢The Nanno Editor◢"
           250  RESTORE 1590
           260  NM=6:GOSUB 1570
           270  GOSUB 1620
           280  IF KEY<49 OR KEY>52 THEN 270
           290  POSITION 23,14:PRINT CHR$(KEY):ON K
                EY-48 GOSUB 320,960,1230,1290
           300  GOTO 240
           310  REM EDIT
           320  TRAP 1780:PRINT "■ESC CTRL <■CMD:";
           330  GOSUB 1627
           340  IF KEY=27 THEN RETURN :REM ESC
           350  GOSUB 1690
           360  FOR J=1 TO 5:IF CHR$(KEY)=CMD$(J,J)
                THEN NEXT J:GOTO 330
           370  PRINT J:CHR$(KEY):IF CHR$(KEY)="A" TH
                EN POSITION 2,0:? "■Enter Address■"
           380  PRINT "■ESC CTRL <■":POSITION 1,2:TRAP 320:PRINT "L#";:I
                NPUT LN:LN=INT(LN):IF LN<1 OR LN>20
                0 THEN 320
           400  TRAP 1780
           410  GOSUB 1700
           420  ON J GOSUB 450,550,610,720,830
           430  GOTO 320
           440  REM ADD
           450  IF NL=200 THEN 500
           460  LN=NL+1
           470  FOR RW=3 TO 21 STEP 2
           480  GOSUB 1700
           490  GOSUB 1330
           500  IF ESC=1 THEN RETURN
           510  NL=LN:LN=LN+1:IF NL=200 THEN 530
           520  NEXT RW:PRINT "■ESC CTRL <■":GOTO 4
                70
           530  PRINT "SOURCE CODE IS A MAXIMUM OF
                200 LINES!":GOSUB 1630:RETURN
           540  REM DELETE
           550  IF LN>NL THEN PRINT "NO SUCH LINE":
                GOSUB 1630:RETURN
           560  PRINT "DELETING LINE ";LN
           570  FOR I=LN TO NL:LINE$(I*76,(I*76)+75
                )=LINE$((I+1)*76,((I+1)*76)+75):NEX
                T I:NL=NL-1:LINE$((NL+1)*76,(NL+1)*76)=
                ""
           580  RETURN
           590  REM EDIT
           600  IF LN>NL THEN PRINT "THERE IS NO LI
                NE ";LN$:GOSUB 1630:RETURN
           610  FOR RW=3 TO 21 STEP 2
           620  GOSUB 1700:POSITION 4,RW:PRINT LINE
                $((LN*76),(LN*76)+75)
           630  GOSUB 1330
           640  IF ESC=1 THEN RETURN
           650  IF NL=LN THEN RETURN
           660  LN=LN+1
           670  NEXT RW
           680  PRINT "■ESC CTRL <■"
           690  GOTO 620
           700  REM INSERT
           710  IF LN>NL THEN PRINT "LAST LINE IS "
                ;NL;" CAN'T INSERT":GOSUB 1630:RETU
                RN
           730  IF NL>=200 THEN PRINT "SOURCE CODE
                IS ALREADY AT MAXIMUM OF 200 LINES.
                ":GOSUB 1630:RETURN
           740  FOR I=NL TO LN STEP -1:LINE$((I+1
                )*76,((I+1)*76)+75)=LINE$(I*76,(I*
                76)+75):NEXT I
           750  NL=NL+1
           760  RW=3:GOSUB 1700:GOSUB 1330
           770  IF ESC=1 THEN 790
           780  RETURN

           790  FOR I=NL TO LN STEP -1:LINE$((I+1)*76
                ,((I+1)*76)+75)=LINE$(I*76,(I*76)+7
                5):NEXT I
           800  NL=NL-1
           810  RETURN
           820  REM LIST
           830  IF LN>NL THEN PRINT "NO SUCH LINE "
                :GOSUB 1630:RETURN
           840  ? "■ESC CTRL <■":FOR RW=3 TO 21 STE
                P 2:POSITION 2,RW:? LINE$((LN*76),(
                LN*76)+75)
           850  GOSUB 1700
           860  PRINT LN$;" ";
           870  PRINT LINE$((LN*76),(LN*76)+75)
           880  LN=LN+1
           890  IF LN>NL THEN 940
           900  NEXT RW:PRINT "■PRESS ESC TO QUI
                T <OR RETURN TO CONTINUE■";
           910  IF PEEK(764)=255 THEN 910
           920  GOSUB 1620:IF KEY=27 THEN RETURN
           930  IF KEY=155 THEN 840
           940  PRINT "■PRESS ANY KEY TO CONTINUE■
                ":GOSUB 1620:RETURN
           950  REM FILES
           960  PRINT "■ESC CTRL <■":POSITION 13,0:
                PRINT "File Device Menu"
           970  RESTORE 1600:NM=5:GOSUB 1570
           980  GOSUB 1620:IF KEY<49 OR KEY>51 THEN
                980
           990  IF KEY=51 THEN RETURN
          1000  IF KEY=50 THEN DEV$(1,2)="C:":GOTO
                1080
          1010  DEV$(1,2)="D:"
          1020  POSITION 1,15:FL$="":PRINT "Input f
                ilename";:INPUT FL$:IF FL$="" THEN
                RETURN
          1030  IF LEN(FL$)>8 THEN FL$(9,LEN(FL$))=
                ""
          1040  IF FL$(1,2)="D:" THEN 1050
          1050  DEV$=FL$:DEV$(LEN(DEV$)+1)=".S"
          1060  DEV$=FL$:FL$(1,2)="D:":FL$(3,2+LEN(
                DEV$))=DEV$
          1070  DEV$=FL$:DEV$(LEN(DEV$)+1)=".S"
          1080  TRAP 1780:PRINT "■ESC CTRL <■":POSITION 13,0:
                PRINT "Load/Save Menu"
          1090  GOSUB 1610:NM=5:GOSUB 1570
          1100  GOSUB 1620:IF KEY<49 OR KEY>51 THEN
                980
          1110  IF KEY=51 THEN RETURN
          1120  IF KEY=50 THEN 1150
          1130  CLOSE #3:IF DEV$="C:" THEN ?:?:"PR
                ESS PLAY/RECORD ON RECORDER THEN PRE
                SS RETURN"
          1140  OPEN #3,8,0,DEV$:GOTO 1200
          1150  CLOSE #3:IF DEV$="C:" THEN ?:?:"PR
                ESS PLAY ON RECORDER THEN PRESS RET
                URN"
          1160  OPEN #3,4,0,DEV$:GOTO 1170
          1170  FOR I=1 TO NL:INPUT #3;A$:LINE$(I*7
                6,(I*76)+75)=A$:NEXT I
          1180  CLOSE #3:RETURN
          1190  CLOSE #3,NL
          1200  FOR I=1 TO NL:PRINT #3;LINE$((I*76),(
                I*76)+75):NEXT I
          1210  CLOSE #3:RETURN
          1220  REM PRINT
          1230  OPEN #4,255:PRINT :PRINT "TURN ON
                PRINTER AND <PRESS ANY KEY>"
          1240  POKE 764,255:PRINT :PRINT "TURN ON
                PRINTER AND <PRESS ANY KEY>"
          1250  GOSUB 1620
          1260  FOR LN=1 TO NL:GOSUB 1700:LPRINT LN
                $;" ";LINE$((LN*76),(LN*76)+75):NEX
                T LN
          1270  RETURN
          1280  REM END PROGRAM
          1290  PRINT "ARE YOU SURE YOU WANT TO END
                THIS SESSION? (Y/N) ";
          1300  GOSUB 1620:IF KEY<>89 AND KEY<>78 T
                HEN 1300
          1310  PRINT CHR$(KEY):IF KEY=89 THEN GRAP
                HICS 0:TRAP 1970:END
          1320  RETURN
          1330  REM INPUT LINE ROUTINE
          1340  IF B=0:ESC=0:POSITION 0,RW:PRINT LN$;
                " ";
          1350  X=(40*RW)+4
          1360  POSITION X-(INT(X/40))*40,INT(X/40)
                :GOSUB 1620
          1370  FOR I=1 TO LEN(C$):IF KEY=ASC(C$(I,
                I)) THEN 1410
          1380  NEXT I
          1390  X=X+1:IF X>(RW+2)*40 THEN X=(RW+2)*
                40
          1400  PRINT CHR$(KEY);:GOTO 1360
          1410  IF KEY=126 AND X>(40*RW)+4 THEN X=X
                -1:PRINT CHR$(KEY);:GOTO 1360
```

Continued

NANOEDITOR *Continued*

ATARI 800/800XL/130XE

```
B 1420 IF KEY=155 THEN POKE 752,1:PRINT :GOSUB 1720:POKE 752,0:GOSUB 1730:RETURN
K 1430 IF KEY=27 THEN ESC=1:RETURN
U 1440 IF KEY=30 AND X>(40*RW)+4 THEN X=X-1:PRINT CHR$(KEY);:IF X=(RW*40)+39 THEN PRINT CHR$(28);
K 1450 IF KEY=31 AND X<40*(RW+2) THEN X=X+1:PRINT CHR$(KEY);:IF X=(RW+1)*40 THEN PRINT CHR$(29);
Z 1460 IF KEY=28 AND X-40>(40*RW)+4 THEN X=X-40:PRINT CHR$(KEY);
V 1470 IF KEY=29 AND X+40<(RW+2)*40 THEN X=X+40:PRINT CHR$(KEY);
L 1480 IF KEY=156 THEN POSITION 4,RW:FOR I=1 TO 76:PRINT " ";:NEXT I:GOTO 1350
A 1490 IF KEY=254 THEN PRINT CHR$(254);
Q 1500 IF KEY<>127 THEN 1550
K 1510 TB=TB+1:IF TB>2 THEN TB=1
A 1520 IF TB=1 THEN X=(40*RW)+11
J 1530 IF TB=2 THEN X=40*(RW+1)
A 1540 POSITION X-INT(X/40)*40,INT(X/40):PRINT CHR$(31);CHR$(30);
Q 1550 GOTO 1360
I 1560 REM PRINT MENU
Z 1570 FOR I=1 TO NM:READ J,A$:POSITION J,(I*2)+2:PRINT A$;:NEXT I
E 1580 RETURN
N 1590 DATA 14,(1) Edit,14,(2) Files,14,(3) Print,14,(4) Exit,0,,11,your choice:
C 1600 DATA 14,(1) Disk (D1:),14,(2) Cassette (C:),14,(3) Exit,0,,11,your choice:
J 1610 DATA 14,(1) Save,14,(2) Load,14,(3) Exit,0,,11,your choice:
E 1620 GET #5,KEY:RETURN
W 1630 FOR IT=1 TO 500:NEXT IT:RETURN
P 1640 RESTORE 1670
Q 1650 FOR I=1 TO 15:READ A:C$(I,I)=CHR$(A):NEXT I
I 1660 CMD$(1,5)="ADEIL"
W 1670 RETURN
U 1680 DATA 155,27,127,126,28,29,30,31,125,255,253,158,254,156,157
R 1690 FOR I=1 TO 10:SOUND 0,15,10,13:NEXT I:SOUND 0,0,0,0:RETURN
D 1700 LN$=STR$(LN):S$=LN$:LN$="000":LN$(4-LEN(S$),LEN(LN$))=S$:RETURN
T 1710 RETURN
P 1720 POSITION 4,RW:FOR I=0 TO 75:GET #4,CH:LINE$((LN*76)+I,(LN*76)+I)=CHR$(CH):NEXT I:RETURN
X 1730 FOR I=0 TO 75
I 1740 IF LINE$((LN*76)+I,(LN*76)+I)<>" " THEN 1770
Z 1750 NEXT I
H 1760 ESC=1
H 1770 RETURN
Y 1780 REM ERROR
N 1790 X=PEEK(195):Y=PEEK(187)*256+PEEK(186):POSITION 1,22:IF X=144 THEN PRINT "◩DISK I/O ERROR◩":GOTO 1950
S 1800 IF X<160 OR X>171 THEN 1940
J 1810 ON (X-159) GOTO 1820,1830,1840,1850,1860,1870,1880,1890,1900,1910,1920,1930
J 1820 PRINT "◩DRIVE NUMBER UNKNOWN◩":GOTO 1950
P 1830 PRINT "◩TOO MANY FILES OPEN◩":GOTO 1950
Q 1840 PRINT "◩DISK FULL◩":GOTO 1950
J 1850 PRINT "◩UNRECOVERABLE I/O ERROR◩":GOTO 1950
F 1860 PRINT "◩FILE NUMBER MISMATCH◩":GOTO 1950
L 1870 PRINT "◩BAD FILE NAME◩":GOTO 1950
W 1880 PRINT "◩POINT DATA LENGTH ERROR◩":GOTO 1950
R 1890 PRINT "◩FILE LOCKED◩":GOTO 1950
J 1900 PRINT "◩UNKNOWN I/O COMMAND◩":GOTO 1950
S 1910 PRINT "◩DIRECTORY FULL◩":GOTO 1950
A 1920 PRINT "◩FILE NOT FOUND◩":GOTO 1950
V 1930 PRINT "◩POINT INVALID◩":GOTO 1950
U 1940 PRINT "◩ERROR NUMBER◩ ";X;" ◩OCCURRED IN LINE ◩";Y
T 1950 GOSUB 1630
T 1960 POP :TRAP 1780:GOTO 240
```

HCM

NANOASSEMBLER

COMMODORE 64

```
N 100 REM ******************
Y 110 REM * NANOASSEMBLER *
V 120 REM ******************
M 130 REM COPYRIGHT 1985
X 140 REM EMERALD VALLEY PUBLISHING CO
Z 150 REM BY ROGER WOOD
L 160 REM AND THE HCM STAFF
K 170 REM HOME COMPUTER MAGAZINE
F 180 REM VERSION 5.6.1
Y 190 REM C-64 BASIC
H 200 CL$=CHR$(147):CR$=CHR$(13):POKE53280,6:POKE53281,12:POKE646,11
A 210 DIMOP$(18):DIMNN(18)
U 220 DIMLI$(200):DIMLB$(25):DIM AD(255)
X 230 DC=1:GOSUB1730:PRINTCL$:RO=5:CO=12:TE$="THE NANOASSEMBLER"
W 240 GOSUB1770:RO=9:CO=1:TE$="LOAD SOURCE FILE FROM:":SL=2:GOSUB1770
E 250 RO=11:CO=14:TE$="(1) DISK":GOSUB1770:RO=13:CO=14:TE$="(2) CASSETTE"
D 260 GOSUB1770
U 270 RO=17:CO=7:TE$="YOUR CHOICE:":GOSUB1770
T 280 GOSUB1700:GOSUB1720:DC=K-48:IF(K<49)OR(K>50)GOTO280
P 290 GOSUB960:IFK<>49ORFL$="" THENRUN
Z 300 GOSUB1210:GOSUB390
N 310 PRINTCL$:RO=5:CO=12:TE$="THE NANOASSEMBLER":GOSUB1770
H 320 RO=9:CO=1:TE$="SAVE OBJECT FILE TO:":SL=1:GOSUB1770:RO=11:CO=14
S 330 TE$="(1) DISK":GOSUB1770:RO=13:CO=14:TE$="(2) CASSETTE":GOSUB1770
F 340 RO=17:CO=7:TE$="YOUR CHOICE:":GOSUB1770:GOSUB1700:GOSUB1720:DC=KE-48
U 350 IF(K<49)OR(K>50)GOTO340
C 360 IFFL$=""THEN310
X 370 GOSUB950:IFK=50GOTO310
C 380 GOTO1250
J 390 PRINT"EXECUTING FIRST PASS":CA=0:LB=0:FORLN=1TONL:LF=0:GOSUB910
J 400 IFMID$(TL$,1,6)="      "GOTO570
U 410 IFLB=24GOTO820
K 420 LB$(LB)=MID$(TL$,1,6):X=0:FORIT=1TO6
D 430 IFCHR$(32)=MID$(LB$(LB),IT,1)THENX=IT:GOTO450
B 440 NEXT
H 450 IFX=0GOTO490
S 460 FORIT=6TOXSTEP-1:IFMID$(LB$(LB),IT,1)=CHR$(32)GOTO480
N 470 GOTO810
E 480 LB$(LB)=MID$(LB$(LB),1,IT-1):NEXT:IFLB=0GOTO510
V 490 FORIT=0TOLB-1:IFLB$(LB)=MID$(LB$(IT),2,LEN(LB$(IT)))GOTO830
W 500 NEXT
F 510 IF(ASC(LB$(LB))>64)AND(ASC(LB$(LB))<91)GOTO530
H 520 GOTO810
W 530 FORIT=1TOLEN(LB$(LB)):X=ASC(MID$(LB$(LB),IT,1)+CHR$(0))
O 540 IF((X>64)AND(X<92))OR((X>47)AND(X<58))GOTO560
G 550 GOTO810
Q 560 NEXT:LF=1
L 570 IFMID$(TL$,7,1)<>" "GOTO800
M 580 GOSUB1170:IFCA+NN(OP)>256 THEN860
F 590 ONOP+1GOTO1260,1360,1330,1330,1260,1260,1260,1260,1260,1260,1260,1330,1330
F 600 ONOP-12GOTO1330,1330,1330,1280,1300,1380
N 610 IFLF=0GOTO630
F 620 LB$(LB)=CHR$(CA)+LB$(LB):LB=LB+1
N 630 CA=CA+NN(OP)
L 640 NEXT:PRINT"EXECUTING SECOND PASS":CA=0:FORLN=1TONL:GOSUB910:GOSUB1170
F 650 EA=CA+NN(OP)
D 660 IFOP=17GOTO740
M 670 IFOP<>18GOTO690
Y 680 GOSUB1480:EA=V:GOTO740
S 690 IFNN(OP)<3GOTO750
Z 700 IFAD(CA+1)<>-1GOTO750
T 710 FORIT=0TOLB:IFMID$(TL$,X+1,LL)=MID$(LB$(IT),2,LEN(LB$(IT)))GOTO730
M 720 NEXT:GOTO850
B 730 V=ASC(LB$(IT)):AD(CA+1)=V-INT(V/16)*16:AD(CA+2)=INT(V/16):GOTO750
S 740 PRINT#PR,"          ";LN;LI$(LN):GOTO770
B 750 PRINT#PR,"          ";LN;LI$(LN):FORIT=0TONN(OP)-1:GOSUB880:PRINT#PR,OT$
E 760 NEXT
A 770 CA=EA:NEXT:PRINT"PRESS ANY KEY TO CONTINUE":GOSUB1700:IFPR=0GOTO790
O 780 CLOSE1
K 790 RETURN
N 800 PRINT#PR,"ILLEGAL CODE IN LINE";LN:GOTO870
T 810 PRINT#PR,"ILLEGAL LABEL IN LINE";LN:GOTO870
T 820 PRINT#PR,"TOO MANY LABELS--   25 MAXIMUM":GOTO870
A 830 PRINT#PR,"DUPLICATE LABEL IN LINE";LN:GOTO870
R 840 PRINT#PR,"NUMBER OUT OF RANGE IN LINE";LN:GOTO870
H 850 PRINT#PR,"ILLEGAL USE OF LABEL IN LINE";LN:GOTO870
S 860 PRINT#PR,"ADDRESS OUT OF RANGE IN LINE";LN:GOTO870
```

Continued

TYPE-IN LISTINGS

NANOASSEMBLER *Continued*

COMMODORE 64

```
 Y  870  GOSUB1720:GOTO1250
 TY 880  OT$=MID$(STR$(CA+IT),2)
 TY 890  OT$=MID$(" 000",1,3-LEN(OT$))+OT$:OU
         $=MID$(ST$(AD(CA+IT)),2)
 C  900  OU$=MID$(" 000",1,3-LEN(OU$))+OU$:OT
         $=OT$+OU$:RETURN
 C  910  LT$=MID$(LI$(LN),1,24):FORIT=LEN(TL
         $)TO1STEP-1
 J  920  IFMID$(TL$,IT,1)<>" "GOTO940
 JOV 930 TL$=MID$(TL$,1,LEN(TL$)-1):NEXT
 LA 940  LI=INTCLR:RETURN
 PA 950  LR$=STR$(CO=9:CO=14:TE$="(1) SAVE":
         GOSUB1770:GOTO970
 V  960  PRINTCL$:RO=9:CO=14:TE$="(1) LOAD":
         GOSUB1770
 E  970  RO=11:CO=14:TE$="(2) EXIT":GOSUB177
         0:ROSUB1770:GOSUB1780:K=KE:IFK<>49GOT
         O1000
 Z  990  ONSLGOSUB1010,1060:CLOSEDN
 ZLX 1010 RETURN
 LX 1020 FL$="":SA=1:DV=1"SAVE FILE":INPUT"F
         ILENAME";FL$:IFFL$=""THEN1140
 U  1030 IFDC=1THENDV=8:SA=8:FL$="@:"+FL$+",
         W":OPEN15,8,15
 E  1040 OPENN1,DV,SA,FL$:IFDC=1THENGOSUB1150
         :IFETHENGOTO1010PRINT#1,AD(IT):NEXT:CL
 F  1050 OSE1:CLOSE15:GOSUB1140:PRINTR$;TAB(12)"L
 Z  1060 OADFL$="FILE":IFDC=1THEN1140
 X  1070 IFFL$=""THEN1140
 XM 1080 FL$=FL$+",S,R":OPEN15,8,15
 J  1090 FL$=FL$+",DV,SA,FL$:IFDC=1THENGOSUB11
         50:PRINT#1,AD(IT):NEXT:CLOSE1:CLOSE15
 HB 1100 INPUT#1,NL:FORIT=1TONL:LI$(IT)=""
 BR 1110 GETA$:IFA$=CHR$(13)GOTO1130
 PE 1120 LI$(IT)=LI$(IT)+A$:GOTO1110
 N  1130 NEXTIT:CLOSE1:CLOSE15
 N  1140 PRINTCHR$(147)"*** ENTER FILENAME ***
         "
 N  1150 INPUT#15,E,E$:IFETHENPRINT"**"E$"
         ":FORI=1TO1000:NEXT:CLOSE1:CLOSE1
         5
 M  1160 RETURN
 MB 1170 FORIT=0TO18:IFMID$(TL$,8,LEN(OP$(IT
         )))=OP$(IT)GOTO1190
 MP 1180 NEXTIT:GOTO800
 P  1190 IFLEN(OP$(OP))=3THENOP$=OP$+"G"GOTO1180
 X  1200 RETURN"SEND ASSEMBLY TO:":PRINT"1. S
 T  1210 PRINT"2. PRINTER"
 E  1220 GOSUB1680:IFK=50THENPR$="4":GOTO124
         0
 YY 1230 PR$="3"
 YC 1240 OPENPR,VAL(PR$),4:RETURN
 CP 1250 PRINT:END
 MA 1260 IFX<>11GOTO800
 AH 1270 GOSUB1480:IFV=-1GOTO850
 NA 1280 GOSUB1480:IFV=-1GOTO850
 AU 1310 FL$="0"GOTO800
 U  1320 LB$(LB)=CHR$(V)+LB$(LB):LB=LB+1:GOT
         O640
 R  1330 GOSUB1480:AD(CA)=OP:IFV<>-1GOTO1350
```

```
 L  1340 AD(CA+1)=-1:AD(CA+2)=-1:GOTO610
 O  1350 AD(CA+1)=V-INT(V/16)*16:AD(CA+2)=IN
         T(V/16):GOTO610
 F  1360 GOSUB1480:IFV=-1GOTO850
 M  1370 AD(CA)=OP:AD(CA+1)=V:GOTO610
 Y  1380 GOSUB1480:IF(V=-1)OR(LF=1)GOTO850
 Z  1390 CA=V:GOTOT640
 Z  1400 V=0:FORIT=0TOLEN(T$)-1:V=V+VAL((MID$
         (T$,IT+1,1))*2!(LEN(T$)-IT-1)):NEXT
 J  1410 IF((OP=16))AND(V>15))OR(V>255)GOTO840
 F  1420 RETURN
 J  1430 V=0:FORIT=0TOLEN(T$)-1:J=0:FORI=1TO
         LEN(HX$)
 Y  1440 IFMID$(T$,IT+1,1)=MID$(HX$,I,1)THEN
         J=I-1
 A  1450 NEXT:V=V+16!(LEN(T$)-IT-1)*J:NEXT
 K  1460 IF((OP=16))AND(V>15))OR(V>255)GOTO840
 U  1470 RETURN
 J  1480 IFMID$(TL$,X,1)<>""GOTO840
 X  1490 IFMID$(TL$,X+1,1)=""GOTO800
 J  1500 Y=0:FORI=1TOLEN(NM$):IFY>0THEN I=LE
         N(NM$):GOTO1520
 J  1510 IFMID$(TL$,X+1,1)=MID$(NM$,I,1)THEN
         Y=I
 H  1520 NEXT
 NV 1530 IFY<>0GOTO1550
 V  1540 RETURN
 Q  1550 IFY<>2GOTO1570
 J  1560 ONYGOTO1580,1600,1660
 J  1580 T$="":FORIT=X+2TOLL:N$=(MID$(TL$,IT
         ,1)):IF(N$<"0")OR(N$>"1")GOTO840
 Q  1590 T$=T$+N$:NEXT:GOTO1400
 NN 1600 T$="":FORIT=X+2TO LL
 N  1610 N$=(MID$(TL$,IT,1)):J=0:FORI=1TOLEN
         (HX$):IF N$=MID$(HX$,I,1)THEN J=I
 S  1620 NEXT
 R  1630 IFJ=0GOTO840
 RN 1640 T$=T$+N$:NEXT:GOTO1430
 N  1650 GOTO1610
 X  1660 T$="":FORIT=X+1TOLL:N$=(MID$(TL$,IT
         ,1)):IF(N$<"0")+(N$>"9")GOTO840
 S  1670 T$=T$+N$:NEXT:V=VAL("%"T$):RETURN
 J  1680 GOSUB1780:K=KE:IF(K<>49)AND(K<>50)G
         OTO1680
 U  1690 PRINTCHR$(K):RETURN
 W  1700 GOSUB1780:IFKE>132THENRETURN
 V  1710 K=KE:POKE(98+(18*40))+20),K:POKE(55
         +(18*40))+20),1:RETURN
 HV 1720 FORIT=0TO250:NEXT:RETURN
 V  1730 FORIT="0"TO"18":READOP$(IT),NN(IT):NEXT
         :HX$="0123456789ABCDEFU"
 Q  1740 NM$="%$012345678 9"":RETURN
 A  1750 DATA ADD,1,LDA#,2,LDA,3,STA,3,TAB,1,
         TBA,1,RRC,1,RLC,1,AND,1,OR,1,XOR,1
 O  1760 DATA ABZ,3,BNZ,3,BCS,3,BCC,3,JMP,3,DN
         ,1,EQU,0,ORG,0
 A  1770 POKE782,CO:POKE781,RO:POKE783,(PEEK
         (783))AND254):SYS65520:PRINTTE$:RETU
         RN
 B  1780 POKE198,0:WAIT197,64:WAIT197,64,64:
         GETKE$:KE=ASC(KE$+CHR$(0)):RETURN
 A  1790 FORI=0TO23:POKEZ+I,0:NEXT:POKEZ,240
         :POKEZ+1,33:POKEZ+2,1:POKEZ+3,1:POKE
         Z+4,129:POKEZ+24,79:FORI=0TO25:NEXT:
 A  1800 POKEZ+4,128:RETURN
```

HCM

NANOEDITOR

COMMODORE 64

```
 T  100  REM ***********
 TDN 110 REM * NANOEDITOR *
 NM 120  REM ***********
 MX 130  REM COPYRIGHT 1985
 L  140  REM EMERALD VALLEY PUBLISHING CO
 ZL 150  REM BY ROGER WOOSTAFF
 K  160  REM AND THE HCM STAFF
 F  170  REM HOME COMPUTER MAGAZINE
 FY 180  REM VERSION 5.6.1
 U  190  REM C-64 BASIC
 U  200  CL$=CHR$(147):CR$=CHR$(13):POKE5328
         0,6:POKE53281,12:POKE646,11:Z=54272
 F  210  FORI=0TO24:POKEZ+I,32:POKEZ+40,160:
         POKEZ+54,15:POKEZ+55,PEEK(Z+5),9:POK
         EZ+6,9:POKEZ+24,15:DIMLI$(202),FL(4)
 T  220  FL(1)=6:FL(2)=33:FL(3)=56:FL(4)=466
 TN 230  PRINTCL$:RO=5:CO=12:TE$="THE NANOED
         ITOR"
 YH 240  RO=7:CO=14:TE$="(1) EDIT":GOSUB1950
 H  250  RO=9:CO=14:TE$="(2) FILES":GOSUB195
         0:RO=11:CO=14:TE$="(3) PRINT":GOSUB
         1950:RO=13:CO=14:TE$="(4) EXIT":GOSUB195
 E  260  RO=15:CO=7:TE$="YOUR CHOICE:"
 A  270  GOSUB1950:TE$="":GOSUB1960:K=KE:IF(
         K<49)OR(K>52)GOTO270
 E  280  PRINTKE$:GOSUB1380:ONK-48GOSUB300,9
         60,1240,1260:GOTO230
 NX 290  REM EDIT
 NX 300  PRINTCL$:LN$="CMD":L=1:RW=1:CN$=MID
         $(C$,LT=157:LN=0:GOSUB1420:LT=0:IFES
         =0THEN330
 Z  320  RETURN
```

```
 U  330  OP=0:J=LEN(LI$(LN)):FORI=1TOLEN(CM$
         ):IFOP=ITHENNEXT:GOTO360
 XJ 340  IFLI$(LN)=MID$(CM$,I,J)THENOP=I
 JH 350  NEXT
 H  360  IF(LI$(LN)=""OR(OP=0)GOTO310
 QR 370  IFOP=1GOTO450
 CN 380  N$=MID$(C$,1,5):L$="L"+CHR$(99)+C
         HR$(127):LN$=L$:L=3:RW=2
 G  390  LI$(0)="":LN=0:GOSUB1420:L=0:IFES=0
         GOTO410
 WQ 400  RETURN
 QL 410  IFLEN(LI$(0))=0GOTO390
 L  420  LN=VAL(LI$(0)):IF(LN>0)AND(LN<201)G
         OTO440
 VI 430  ES$="":GOSUB1950:GOTO390
 VI 440  LN$=MID$("000",1,3-LEN(LI$(0)))+LI$
         (0)
 N  450  ONOPGOSUB470,560,620,710,790:GOTO30
         0
 FC 460  REM ADD
 C  470  IFNL=200GOTO530
 IG 480  FORRW=3TO21STEP2:GOSUB1320:LT=0:GOS
         UB1420:POKE204,1:IFES=0GOTO510
 RW 500  RETURN
 RW 510  NL=LT:LN=LN+1:IFNL=200GOTO540
 W  520  NEXT
 IA 530  PRINTCL$:GOTO490
 B  540  PRINT"SOURCE CODE IS A MAXIMUM OF 2
         00 LINES":GOSUB1380:RETURN
 BX 550  REM DELETE
 F  560  IFLN<=NLGOTO580
 F  570  PRINT" NO SUCH LINE":GOSUB1380:RETU
         RN
```

Continued

```
C 580 PRINT"DELETING LINE";LN:FORIT=LNTON
L:LI$(NL+1):NEXT:NL=NL-1
T 590 LI$(IT)=LI$(IT+1):
TOX 600 REM END PROGRAM
XRG 610 REM MID$(STR$(LN),2):LN$=(MID$("000
IFLN<"TGOTO640
Q 630 PRINT"THERE IS NO LINE ";LN$:GOSUB1
380:RETURN
B 640 FORRW=3TO21STEP2:GOSUB1320:X=0:N=0:
ES=0:GOSUB1360:TE$=LI$(LN):GOSUB190
0
Q 650 GOSUB142O:IFES=0GOTO670
KT 660 LI$(LN)=E$:RETURN
T 670 IFLEN(LI$(LN))=0GOTO580
BA 680 IFNL=LNTHENGOTO600
A 690 LN=LN+1:NEXT:PRINTCL$:GOTO640
U 700 REM INSERT
UEC 710 IF(LN<=NL)ANDN<LN<200):GOTO730
C 720 PRINTSPC(LAST LINE IS";NL;"CAN'T INSER
T"):GOSUB1380:RETURN
A 730 RW=3:GOSUB1320:GOSUB1360:S$=LI$(LN)
H 740 LT=0:E=0:GOSUB1420:IFES=0GOTO760
HAJ 750 LI$(LN)=S$:RETURN
JL 760 IFLEN(LI$(LN))=0GOTO740
F 770 FORIT=NLTOLNSTEP-1:LI$(IT+1)=LI$(IT
):NEXT:LI$=S$:RETURN
N 780 REM LIST
NYA 790 IFLN<=630LGOTO810
AL 800 GOSUB1320:GOSUB1360:TE$=LI$(LN):GOS
UB1420:IFES=0GOTO760:GOTO800
L 810 PRINTCL$;:FORRW=3TO21STEP2:S$=LI$(LN
):GOSUB1320:C=7:X=0:N=0:TE$=S$:GOSUB
1360:GOSUB1420:PRINTLN;:IFES=0GOTO81
0:PRINTSPC(2);:GOSUB1320:C=1:X=0:GOSU
I 820 B1360:GOSUB1420:PRINTLN;:IFES=0GOTO81
0:PRINTSPC(2);:GOSUB1320:C=1:X=0:GOSU
B1360:GOSUB1420:PRINTLN;:IFES=0GOTO81
A 830 0:PRINTSPC(2);:GOSUB1320:C=8:X=0:GOSU
B1360:GOSUB1420:PRINTLN;:IFES=0GOTO81
0:PRINTSPC(2);:GOSUB1320:C=2:X=0:GOSU
UM 840 B1360:GOSUB1420:PRINTLN;:IFES=0GOTO81
M 850 0:PRINTSPC(2);:GOSUB1320:C=13:X=0:GOS
UB1360:GOSUB1420:PRINTLN;:IFES=0GOTO
FS 1 810:QUIT:ORSPACETOCONTINUE>
GS 860 GOSUBE1950:IFLN<=NLGOTO850
IF 870 GOSUBE1960:IFKE=32THENRETURNELSE860
SM 880 GOTO40
MN 890 KEYT:GOSUBE1960:IFKS=ITHENNEXTRW:GOT
O810
NRZ 900 IFCHR$(K)="DTHENGOSUBE1970:GOTO890
XM 910 NEXT
RZ 920 IF(KS<1)OR(KS>2)GOTO860
XM 930 IFKS=1GOTO810
MF 940 RENUMBER:CALL
F 950 REM INITIALISE:DIMLI$(500):CL$=CHR$(
147):RETURN
P 960 PRINTSPC(5)"DELETE MENU":PRINT:PRINT"
1 - DELETE A LINE":PRINT"2 - DELETE
R 970 BLOCK":PRINT"3 - QUIT":PRINT:INPUT"
SELECT";TE$:IFTE$="1"THENGOSUB1950:G
OTO1450
K 980 IFTE$="2"THENGOSUB1950:GOSUB1970:GOT
O1450
P 990 IFTE$="3"THENPRINTCL$:END
R 1000 GOTO960
X 1010 PRINTCL$:PRINT"DELETE BLOCK":PRINT:P
RINT"START LINE";:GOSUB1950:LS=LN:PRI
H 1020 NT"END LINE";:GOSUB1950:LE=LN:IFLE<LS
THENLN=LE:LE=LS:LS=LN
D 1030 IFLS=LETHENLN=LS:GOSUB1950:GOTO1450
M 1040 FORLN=LSTOLE:LI$(LN)="":NEXT:GOTO145
0
M 1050 REM*****SAVE*****:GOSUB1970:GOSUB1
MZ 1060 120:CLOSE1:RETURN
A 1070 REM SAVE:PRINT"SAVE FILE NAME";:GOS
UB1120:OPEN1,1,1,FL$:FORLN=1TONL:PRI
NT#1,LI$(LN):NEXT:CLOSE1:RETURN
F 1080 REM LOAD:PRINT"LOAD FILE NAME";:GOSU
H 1090 B1120:OPEN1,1,0,FL$:INPUT#1,NL:FORLN=
1TONL:LI$(LN)="":INPUT#1,LI$(LN):NEX
T:CLOSE1:RETURN
M 1100 REM VERIFY:PRINT"VERIFY FILE NAME";:G
OSUB1120:OPEN1,1,0,FL$:FORLN=1TONL:I
F 1110 LI$(LN)<>LEFT$(CL$,LEN(LI$(LN)))THENP
RINT"VERIFY ERROR":CLOSE1:GOTO1450
B 1120 NEXT:CLOSE1:PRINT"VERIFIED":GOTO1450
M 1130 REM DIR:PRINT"DIRECTORY":OPEN1,8,0,"$
VL 1140 0":GET#1,A$,A$:GET#1,A$,A$:IFST<>0TH
ENCLOSE1:GOTO1450
I 1150 GET#1,A$,A$:IFA$=CHR$(0)THENPRINT:GO
TO1140
NH 1160 PRINT:PRINTA$;:GET#1,A$:IFA$<>CHR$(0
)THENPRINTA$;:GOTO1160
HH 1170 PRINT:GOTO1140
HH 1180 CLOSE1:GOTO1450
HX 1190 REM*****TEXT*****:GOSUB1970:GOSUB
W 1200 1120:OPEN1,1,1,FL$:FORLN=1TONL:PRINT#1
,LI$(LN):NEXT:CLOSE1:RETURN
N 1210 REM*****MERGE*****:GOSUB1970:GOSUB
1120:OPEN1,1,0,FL$:INPUT#1,NL:FORLN=
NLTONL+NL:INPUT#1,LI$(LN):NEXT:CLOSE1
:RETURN
NA 1220 REM*****PRINT*****:FORIT=1TO2000:LI
R 1230 REM PRINT:PRINTCL$;:RO=5:CO=10:TE$="
RK 1240 PRINTING FROM LINE";:GOSUB1950:GOSUB
1320:PRINT"PRINTER SETUP":PRINT"1 -
P 1250 SINGLE SHEET":PRINT"2 - CONTINUOUS
FEED":PRINT:INPUT"SELECT";TE$:IFTE$<
O 1260 >"1"ANDTE$<>"2"THEN1250
R 1270 OPEN4,4:IFTE$="2"THENPRINT#4,CHR$(10
Y 1280 );:IFK=89GOTO1300
```

```
Y 1290 RETURN
B 1300 REM*****END PROGRAM*****:END
E 1310 REM*****PROGRAM*****
LN$=MID$(STR$(LN),2):LN$=(MID$("000
",3-LEN(LN$))+LN$):RETURN
C 1330 L=48:B=ASC("A"):T=ASC("Z"):CN$=C$:R
ETURN
G 1340 FORIT=3TO5:POKE(983+(RW*40))+IT),ASC
(MID$(LN$,IT-2,1))-64
A 1350 POKE(55255+(RW*40))+IT),1:NEXT:RETUR
N
N 1360 FORIT=3TO5:POKE(983+(RW*40))+IT),ASC
(MID$(LN$,IT-2,1))
Q 1370 POKE(55255+(RW*40))+IT),1:NEXT:RETUR
N
X 1380 FORIT=1TO1000:NEXT:RETURN
JZ 1390 REM*****PROGRAM INIT*****
1400 CS=CHR$(133)+CHR$(20)+CHR$
S 1410 (29)+CHR$(157)+CHR$(134)+CHR$(135)
X 1420 CM$="ADE":RETURN
LI$(0)="":X=0:N=0:ES=0
E 1430 IFLN$=""CMD"ORLN$=LL$THENGOSUB1340:
F 1440 ITES=""":GOSUB1900:GOTO1450
W 1450 GOSUB1360:GETKE$:POKE207,0:IFKE$=""
GOTO1450
Y 1460 POKEZ+4,16:K=ASC(KE$)
I 1470 ON-R((K>47)ANDK<58)OR(K>64)ANDK<91
)ONR((K>34)ANDK<38)OR(K=32))GOTO1540
AG 1480 OIFLEN(CN$)<>1THEN1520
H 1500 IFK=133THENES=1
JC 1520 GON-(K=20)-2*(K=29)-3*(K=148)-4*(K=1
57)GOTO1610,1650,1670,1700
E 1530 ON-(K=133)-2*(K=134)-3*(K=135)GOSUB
1730,1740,1760:ONESGOTO1600:GOTO1450
A 1540 ON-(X=70)GOTO1450:ON-(LT>0)GOTO1910
:ON-(L=3)GOTO1930
H 1550 IFLEN(LI$(LN))=XTHENLI$(LN)=LI$(LN)
+KE$:X=X+1:GOTO1570
W 1560 X=X+1:LI$(LN)=LEFT$(LI$(LN),X-1)+KE
$+RIGHT$(LI$(LN),LEN(LI$(LN))-X)
P 1570 POKEZ+4,17:POKE204,1:TE$=LI$(LN):GO
SUB1950:IFLT>0THENX=X-1
B 1640 GOSUB1950:GOTO1450
NS 1660 IFX=X+1:GOTO1570
SC 1670 IFL=LEN(LI$(LN))=72GOTO1450
K 1680 LIFT$(LI$(LN),X)+CHR$(32)+RIGHT$(LI
$(LN),LEN(LI$(LN))-X)
R 1690 LEFT$(LI$(LN),X):GOSUB1950:GOTO1450
V 1700 IFX=0GOTO1450
N 1710 ITE$=LI$(LN)+CHR$(32):POKE204,1:GOSU
B1950:X=X-1:TE$=LEFT$(LI$(LN),X)
OC 1730 GOSUB1950:GOTO1450
N 1740 TE$="":FORI=1TO72:TE$=TE$+CHR$(32):
NEXT:POKE204,1:GOSUB1950:TE$="":RETURN
K 1750 IFLN$="95"THENX=0:RETURN
H 1760 IFLN$="95"THEN+CHR$(99)+CHR$(127)THEN
RETURN
E 1770 IFX<77THENN=1
EN 1780 IF(X>63)ANDX<34)THENN=2
OE 1790 IF(X>33)ANDX<57)THENN=3
G 1810 IFX>56THENRETURN
TE$=LI$(LN):POKE204,1:GOSUB1950:TE$
U 1820 =TE$+CHR$(32):X=FL(N)+1
H 1830 IFX<=QLENOX-LEN(LI$(LN))GOTO1850
FLN)+CHR$(32):NEXT:TE$=LI$(LN):POKE2
04,1:GOSUB1950:TE$=LI$(LN):POKE2
K 1840 04,1:GOSUB1950
H 1850 GOTE$=LEFT$(LI$(LN),X):GOSUB1950:RETU
RN
A 1860 POKE204,1:TE$=LI$(LN)+CHR$(32):GOSU
B1950
H 1870 J=0:FORI=1TOLEN(LI$(LN)):IFASC(MID$
(LI$(LN),I,1)+CHR$(0))<>32THENJ=1
F 1880 NEXT:IF(J=0)OR(LI$(LN)="")THENLI$(L
N)="":X=0:TE$="":GOSUB1950:GOTO1450
W 1890 RETURN
WW 1900 RON-RW-1:CO=6:X=0:GOSUB1950:RETURN
P 1910 ON-(K=68)-(K=69)-(K=73)-(K=7
6)GOTOHR$(K):GOTO1450
WH 1920 IFK=KHR$(K)ORK>57)ORX=LGOTO1450
HC 1930 KE$=CHR$(K):GOTO1550
CA 1940 KE$=CHR$(K):GOTO1550
1950 POKE782,CO:POKE781,RO:POKE783,0:SYS
65520:PRINTTE$;:RETURN
F 1960 POKE198,0:WAIT197,64:WAIT197,64,64:
GETKE$:KE=ASC(KE$+CHR$(0)):RETURN
```

HCM

TYPE-IN LISTINGS

```
A 100 '  ******************
D 110 '  *  NANOASSEMBLER  *
Y 120 '  ******************
X 130 '  COPYRIGHT 1985
W 140 '  EMERALD VALLEY PUBLISHING CO
M 150 '  BY ROGER WOOD
Q 160 '  AND THE HCM STAFF
Z 170 '  HOME COMPUTER MAGAZINE
Q 180 '  VERSION 5.6.1
J 190 '  DOS 2.1 AND EITHER
           IBM PCjr W/CARTRIDGE BASIC OR
O 200 '  IBM PC W/BASICA OR
D 210 '  TANDY 1000 W/GW BASIC
Z 220 KEY OFF:COLOR 14,1:WIDTH 40:CLS:X=3:Y=5:W=30:H=5:GOSUB 1430:COLOR 5:LOCATE 5,11,0:PRINT"The NanoAssembler":COLOR 15:LOCATE 10,6:PRINT"PLEASE WAIT WHILE I SET UP":FOR D=1 TO 1000:NEXT
N 230 DIM OP$(19),NN(19),LI$(201),LB$(26),AD%(256),LV(26)
K 240 SP$=CHR$(250):HX$="0123456789ABCDEF":NM$="%$0123456789"
S 250 FOR IT=0 TO 18:READ OP$(IT),NN(IT):NEXT
I 260 GOSUB 1260:GOSUB 640:ASM=1:GOSUB 270:GOSUB 1350:GOTO 690
I 270 'ASSEMBLE
A 280 COLOR 14,1:CLS:PRINT"EXECUTING FIRST PASS":COLOR 4,0:CA=0:LB=0
I 290 FOR LN=1 TO NL:GOSUB 780
J 300 IF CA+NN(OP)>256 THEN 620
G 310 ON OP+1 GOTO 710,760,740,740,710,710,710,710,710,710,710,740,740,740,740,740,720,730,770
U 320 IF LFLAG = 1 THEN LB$(LB)=LB$:LV(LB)=CA:LB=LB+1
C 330 CA=CA+NN(OP)
Q 340 NEXT LN
I 350 'SECOND PASS
H 360 COLOR 14,1:PRINT"EXECUTING SECOND PASS":PRINT:CA=0:EA=0
Q 370 FOR LN=1 TO NL
O 380 GOSUB 780:GOSUB 1670:EA=CA+NN(OP)
C 390 IF OP=18 THEN EA=V:GOTO 440:ELSE IF OP=16 THEN 460:ELSE IF OP=17 THEN 440
P 400 IF NN(OP)<3 THEN 460
M 410 IF AD%(CA+1)<>-1 THEN 460
P 420 IT=0:WHILE (IT < LB AND DT$<>LB$(IT)):IT=IT+1:WEND
C 430 IF IT = LB THEN COLOR 4,0:GOTO 610 ELSE V=LV(IT):AD%(CA+1)=V MOD 16:AD%(CA+2)=V\16:GOTO 460
X 440 PRINT#1,TAB(8);LN;TAB(12);A$;TAB(21);B$;TAB(40);
O 450 IF F$="SCRN:" THEN PRINT:PRINT TAB(12);C$:GOTO 520:ELSE PRINT#1,C$:GOTO 520
L 460 PRINT#1,TAB(8);LN;TAB(12);A$;TAB(21);B$;TAB(40);
I 470 IF F$="SCRN:" THEN PRINT:PRINT TAB(12);C$:ELSE PRINT#1,C$
C 480 FOR IT=0 TO NN(OP)-1
N 490 OT$=MID$(STR$(CA+IT),2):OT$=STRING$(3-LEN(OT$),"0")+OT$
I 500 OU$=MID$(STR$(AD%(CA+IT)),2):OU$=STRING$(3-LEN(OU$),"0")+OU$
L 510 PRINT#1,OT$;"  ";OU$:NEXT
V 520 CA=EA:NEXT
L 530 PRINT"PRESS ANY KEY TO CONTINUE"
C 540 K$=INKEY$:IF K$="" THEN 540
Y 550 CLOSE:RETURN
L 560 PRINT #1,"ILLEGAL CODE IN LINE";LN:GOTO 630
X 570 PRINT #1,"ILLEGAL LABEL IN LINE";LN:GOTO 630
X 580 PRINT #1,"TOO MANY LABELS--  25 MAXIMUM":GOTO 630
A 590 PRINT #1,"DUPLICATE LABEL IN LINE";LN:GOTO 630
Q 600 PRINT #1,"NUMBER OUT OF RANGE IN LINE";LN:GOTO 630
A 610 PRINT #1,"ILLEGAL USE OF LABEL IN LINE";LN:GOTO 630
B 620 PRINT #1,"ADDRESS OUT OF RANGE IN LINE";LN:GOTO 630
O 630 FOR D=0 TO 1000:NEXT:GOTO 690
A 640 'GET PRINTER OPTION
D 650 LOCATE 18,3,1:COLOR 15:PRINT"SEND ASSEMBLY TO:":PRINT"    1. SCREEN":PRINT"    2. PRINTER":LOCATE 22,3
Q 660 K$=INKEY$:IF K$="1" OR K$="2" THEN PRINT K$; ELSE 660
P 670 IF K$="2" THEN F$="LPT1:" ELSE F$="SCRN:"
A 680 OPEN "O",1,F$:RETURN
D 690 'END PROGRAM
D 700 PRINT:PRINT:PRINT:ON ERROR GOTO 0:END
D 710 IF DT$<>"" THEN 560 ELSE AD%(CA)=OP:GOTO 320
M 720 IF V=-1 THEN 610 ELSE AD%(CA)=V:GOTO 320
C 730 IF V=-1 THEN 610 ELSE LB$(LB)=LB$:LV(LB)=V:LB=LB+1:GOTO 340
A 740 AD%(CA)=OP:IF V=-1 THEN AD%(CA+1)=-1:AD%(CA+2)=-1:ELSE AD%(CA+1)=V MOD 16:AD%(CA+2)=V\16
M 750 GOTO 320
Z 760 IF V=-1 THEN 610 ELSE AD%(CA)=OP:AD%(CA+1)=V:GOTO 320
T 770 IF V=-1 OR LFLAG = 1 THEN 610 ELSE CA=V:GOTO 340
J 780 'PARSE LINE
I 790 LFLAG=0:V=0:L$=LI$(LN):M1=INSTR(L$,SP$):M2=INSTR(M1+1,L$,SP$)
R 800 IF M1=1 THEN LB$="" ELSE GOSUB 850
C 810 IF M2=M1+1 THEN OP$="":DT$="" RETURN
N 820 M3=INSTR(M1,L$," "):IF M3=0 OR M3>M2 THEN M3=M2
D 830 GOSUB 950:GOSUB 990
Y 840 RETURN
T 850 'DETERMINE LABEL
E 860 M3=M1-1:WHILE (MID$(L$,M3,1)=" " AND M3>1):M3=M3-1:WEND:LB$=LEFT$(L$,M3)
J 870 IF M3=1 THEN IF LB$=" " THEN LB$="":RETURN
D 880 IF ASC(LB$) < 65 OR ASC(LB$) > 90 THEN 570
C 890 FOR IT=1 TO M3:D$=MID$(LB$,IT,1)
Z 900 IF D$=" " OR D$>"Z" OR D$ < "0" OR (D$>"9" AND D$<"A") THEN 570
N 910 NEXT:IF LB=24 THEN 580
X 920 IF LB>0 THEN FOR IT=0 TO LB:IF LB$(LB)=LB$ THEN 590:ELSE NEXT
A 930 LFLAG = 1
B 940 RETURN
M 950 'EVALUATE OPCODE
S 960 OP$=MID$(L$,M1+1,M3-M1-1):OP=0:WHILE (OP<19 AND OP$(OP)<>OP$):OP=OP+1:WEND
F 970 IF OP=19 THEN 560
F 980 RETURN
Y 990 'EVALUATE DATA
A 1000 WHILE (MID$(L$,M3,1)=" "):M3=M3+1:WEND:IF MID$(L$,M3,1)=SP$ THEN DT$="":RETURN
Q 1010 M2=M2-1:WHILE (MID$(L$,M2,1)=" "):M2=M2-1:WEND:DT$=MID$(L$,M3,M2-M3+1)
M 1020 IF DT$="" THEN RETURN
R 1030 Y=INSTR(NM$,LEFT$(DT$,1)):IF Y=0 THEN V=-1:RETURN:ELSE IF Y>2 THEN Y=3
G 1040 ON Y GOTO 1050,1080,1110
D 1050 FOR IT=2 TO LEN(DT$):D=ASC(MID$(DT$,IT,1))-48:IF D<0 OR D>1 THEN 600
J 1060 V=2*V+D:IF V>255 THEN 600
H 1070 NEXT:RETURN
Y 1080 FOR IT=2 TO LEN(DT$):D=INSTR(HX$,MID$(DT$,IT,1))-1:IF D<0 THEN 600
P 1090 V=16*V+D:IF V>255 THEN 600
B 1100 NEXT:RETURN
B 1110 FOR IT=1 TO LEN(DT$):D=ASC(MID$(DT$,IT,1))-48:IF D<0 OR D>9 THEN 600
L 1120 V=10*V+D:IF V>255 THEN 600
K 1130 NEXT:RETURN
N 1140 'TRAP FILE NAME ERROR
P 1150 GOSUB 1170:RESUME 1160
S 1160 IF ASM=1 THEN GOTO 1350 ELSE GOTO 1260
H 1170 'DISPLAY FILE ERROR MESSAGE
Q 1180 COLOR 8,1:CLS:X=7:Y=5:W=30:H=5:GOSUB 1410:Y=Y+4
R 1190 COLOR 0,5:LOCATE X+2,Y,0:PRINT " INVALID FILE NAME";
O 1200 FOR J = 1 TO 4000:NEXT:LOCATE ,,1:COLOR 15,0
S 1210 RETURN
J 1220 'GET FILE NAME
H 1230 COLOR 14,1:CLS:X=5:Y=5:W=30:H=5:COLOR 4,2:GOSUB 1410:COLOR 15,1:LOCATE X+2,(40-LEN(P$))/2,1:PRINT P$;
M 1240 X=11:Y=1:W=39:H=5:COLOR 15:GOSUB 1430:LOCATE 13,3:PRINT"FILE NAME: ";
J 1250 X=13:Y=14:MC$="!":XC$="z":COLOR 14:S$="":L=20:GOSUB 1500:F$=S$:RETURN
Y 1260 'LOAD FILE
A 1270 P$ = "LOAD SOURCE FILE":GOSUB 1230:ON ERROR GOTO 1140
H 1280 OPEN "I", 1, F$ + ".S"
F 1290 INPUT #1,NL:FOR I = 1 TO NL
C 1300 INPUT #1, S$:QT = 1
A 1310 WHILE (QT>0):QT=INSTR(QT,S$,CHR$(2)):IF QT>0 THEN MID$(S$,QT,1)=CHR$(34)
J 1320 WEND        'REPLACE QUOTATION MARKS...
N 1330 LI$(I) = S$:NEXT:CLOSE:LN=1:FOR I=NL+1 TO 200:LI$(I)=SP$+SP$:NEXT
Q 1340 ON ERROR GOTO 0:RETURN
Z 1350 'SAVE FILE
V 1360 CLS:WIDTH 40:P$ = "SAVE OBJECT FILE"
A 1370 GOSUB 1220:ON ERROR GOTO 1140
K 1380 OPEN "O", 1, F$
E 1390 FOR IT=0 TO 248 STEP 8:WRITE #1,AD%(IT),AD%(IT+1),AD%(IT+2),AD%(IT+3),AD%(IT+4),AD%(IT+5),AD%(IT+6),AD%(IT+7):NEXT:CLOSE
N 1400 RETURN
N 1410 'DISPLAY A SINGLE-SIDED BOX
R 1420 UL=218:UR=191:LL=192:LR=217:H1=196:H2=H1:V1=179:V2=V1:GOTO 1450
Q 1430 'DISPLAY A DOUBLE-SIDED BOX
E 1440 UL=201:UR=187:LL=200:LR=188:H1=205:H2=H1:V1=186:V2=V1:GOTO 1450
```

Continued

NANOASSEMBLER *Continued*

IBM PC/PCjr, TANDY 1000

```
W 1450  '     DISPLAY BOX(0,0,24,79,2,7,0);:LOCATE 1,2:PRINT"NANOASSEMBLER";:LOCATE 2,2:PRINT"Version 1.0";
H 1460  'LOCATE 2,CINT(40-LEN(CHR$(ULR)+"Written by Richard Mansfield"+CHR$(ULR))/2):PRINT"Written by Richard Mansfield";
U 1470  'LOCATE 3,CINT(40-LEN(CHR$(ULR)+"Machine Language Editor/Assembler"+CHR$(ULR))/2):PRINT"Machine Language Editor/Assembler";
A 1480  'LOCATE 4,CINT(40-LEN(CHR$(ULR)+"Version 1.0"+CHR$(ULR))/2):PRINT"Version 1.0";
F 1490  'LOCATE 5,CINT(40-LEN(CHR$(ULR)+"Written by Richard Mansfield"+CHR$(ULR))/2):PRINT"Written by Richard Mansfield";
X 1500  'RETURN
D 1510  'INPUT"Start at line number";X
I 1520  'IF X<0 THEN X=0
X 1530  'K$="":PRINT"Enter lines of code. Type ? to see options.":PRINT
X 1540  'K$=INKEY$:IF K$="" GOTO 1540
O 1550  'IF K$="?" THEN GOSUB 1650:GOTO 1540
X 1560  'IF K$="Q" THEN END
Y 1570  'IF K$="B" THEN GOSUB 1650:GOTO 1540
Q 1580  'IF K$="R" THEN RETURN
Y 1590  'IF K$="L" THEN LIST:GOTO 1540
D 1600  'IF K$="S" THEN GOSUB 1650:INPUT"Save to what filename";FILE$:OPEN"O",1,FILE$:LIST#1,0-X:CLOSE:GOTO 1540
O 1610  'IF K$="M" THEN GOSUB 1650:INPUT"Move from what line";FROM:INPUT"Move to what line";TO:INPUT"How many lines";HOW:LIST FROM-HOW:DELETE FROM-HOW:LIST TO-1:DELETE TO-1:GOTO 1540
W 1450  'IF K$="D" THEN GOSUB 1650:INPUT"Delete what line";DEL:DELETE DEL:GOTO 1540
H 1460  'IF K$="C" THEN GOSUB 1650:INPUT"Copy what line";FROM:INPUT"Copy to what line";TO:INPUT"How many lines";HOW:LIST FROM-HOW:DELETE FROM-HOW:LIST TO-1:DELETE TO-1:GOTO 1540
U 1470  'IF K$="F" THEN GOSUB 1650:INPUT"Find what";FIND$:LIST X-1:GOTO 1540
A 1480  'IF K$="N" THEN GOSUB 1650:INPUT"Find next";FIND$:LIST X-1:GOTO 1540
F 1490  'IF K$="A" THEN GOSUB 1650:INPUT"Change what";FIND$:INPUT"Change to what";CHANGE$:LIST X-1:GOTO 1540
X 1500  'IF K$="T" THEN GOSUB 1650:INPUT"Trace from what line";TRACE:LIST TRACE-1:GOTO 1540
D 1510  'IF K$="P" THEN GOSUB 1650:INPUT"Print from what line";PRINTFROM:INPUT"Print to what line";PRINTTO:LPRINT"Program listing":LIST#1,PRINTFROM-PRINTTO:GOTO 1540
I 1520  'IF K$="I" THEN GOSUB 1650:INPUT"Insert before what line";INSERT:LIST INSERT-1:GOTO 1540
X 1530  'IF K$="E" THEN GOSUB 1650:INPUT"Edit what line";EDIT:LIST EDIT-1:GOTO 1540
X 1540  'IF K$="U" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
O 1550  'IF K$="O" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
X 1560  'IF K$="H" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
Y 1570  'IF K$="W" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
Q 1580  'IF K$="Y" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
Y 1590  'IF K$="Z" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
D 1600  'IF K$="V" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
O 1610  'IF K$="G" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
```

```
N 1620  'IF K$="K" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
K 1630  'IF K$="J" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
P 1640  'IF K$="X" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
W 1650  'IF K$="S" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
V 1660  'IF K$="C" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
N 1670  'IF K$="V" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
V 1680  'IF K$="B" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
P 1690  'IF K$="N" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
Y 1700  'IF K$="M" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
L 1710  'IF K$="," THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
O 1720  'IF K$="." THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
W 1730  'IF K$="/" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 1540
Q 1740  'DATA ADD,1,LDA#,2,LDA,3,STA,3,TAB,1
    DATA ADD,3,RC,1,RLC,1,AND,1,OR,1,XOR,1
    DATA ADC,3,BZ,3,BNZ,3,BCS,3,BCC,3,JMP,3,D
    DATA INC,1,DEC,1,TAX,1,INX,1,DEX,1,TXA,1
    DATA CLC,1,SEC,1,CLI,1,SEI,1,NOP,1,RTS,1
    DATA PHA,1,PLA,1,PHP,1,PLP,1,BRK,1
    DATA ASL,1,ASL,2,ROL,1,ROL,2,LSR,1,LSR,2
    DATA ROR,1,ROR,2,STX,2,STX,3,LDX#,2,LDX,2
    DATA LDX,3,STY,2,STY,3,LDY#,2,LDY,2,LDY,3
    DATA CPY#,2,CPY,2,CPY,3,CMP#,2,CMP,2,CMP,3
    DATA CPX#,2,CPX,2,CPX,3,SBC#,2,SBC,2,SBC,3
    DATA EOR#,2,EOR,2,EOR,3,ORA#,2,ORA,2,ORA,3
    DATA JSR,3,STA,2,STA,3,LDA#,2,LDA,2,LDA,3
    DATA ,0
    'FOR I=1 TO 256:READ A$,B:IF A$="" THEN 1750
    'A$=A$+CHR$(32):A$=LEFT$(A$,4):OP$(I)=A$:LN(I)=B:NEXT
    'RETURN
    'FOR I=1 TO 256:IF OP$(I)=A$ THEN RETURN ELSE NEXT
    'RETURN
    'IF K$="" THEN RETURN
    'IF K$="0" THEN K$="A"
    'IF K$="1" THEN K$="B"
    'IF K$="2" THEN K$="C"
    'IF K$="3" THEN K$="D"
    'IF K$="4" THEN K$="E"
    'IF K$="5" THEN K$="H"
    'IF K$="6" THEN K$="L"
    'IF K$="7" THEN K$="M"
    'RETURN
```

NANOEDITOR

IBM PC/PCjr, TANDY 1000

```
S 100  ''*** NANOEDITOR ***
U 110  ''*** COPYRIGHT 1985 ***
C 120  ''*** RICHARD MANSFIELD ***
X 130  ''*** PUBLISHING CO ***
W 140  ''*** ALL RIGHTS RESERVED ***
M 150  ''*** WORLDWIDE ***
Q 160  ''*** HOME COMPUTER MAGAZINE ***
Z 170  ''*** VERSION 5.6.1 ***
Q 180  ''*** 5.2.1 ***
J 190  ''*** AND UP ***
O 200  ''*** IBM PC ***
D 210  ''*** IBM PCjr/TANDY 1000 ***
L 220  CLEAR 40000:KEY OFF:WIDTH 40:COLOR 14,1,0:CLS:PRINT"NANOEDITOR":PRINT"Version 5.6.1":PRINT"Copyright 1985 Richard Mansfield":PRINT"All rights reserved worldwide":PRINT"Home Computer Magazine":PRINT"IBM PC/PCjr/TANDY 1000":PRINT"Basic 2.1 and up":PRINT"Either 40 or 80 column screen":PRINT"Color or monochrome":PRINT"Disk or cassette"
H 230  DEFINT A-Z:DIM OP$(256),LN(256):FOR I=1 TO 256:READ A$,B:IF A$="" THEN 1750
R 240  A$=A$+"  ":A$=LEFT$(A$,4):OP$(I)=A$:LN(I)=B:NEXT:RETURN
H 250  FOR I=1 TO 256:IF OP$(I)=A$ THEN RETURN ELSE NEXT:RETURN
Z 260  PRINT"Enter lines of code. Type ? to see options.":PRINT:INPUT"Start at line number";X:IF X<0 THEN X=0
V 310  K$="":PRINT:PRINT"Enter lines of code. Type ? to see options.":PRINT:K$=INKEY$:IF K$="" GOTO 310
N 320  IF K$="?" THEN GOSUB 1650:GOTO 310
W 330  IF K$="Q" THEN END
E 340  IF K$="B" THEN GOSUB 1650:GOTO 310
E 350  IF K$="L" THEN LIST:GOTO 310
AR 360  IF K$="S" THEN GOSUB 1650:INPUT"Save to what filename";FILE$:OPEN"O",1,FILE$:LIST#1,0-X:CLOSE:GOTO 310
RH 380  IF K$="M" THEN GOSUB 1650:INPUT"Move from what line";FROM:INPUT"Move to what line";TO:INPUT"How many lines";HOW:LIST FROM-HOW:DELETE FROM-HOW:LIST TO-1:DELETE TO-1:GOTO 310
P 390  IF K$="D" THEN GOSUB 1650:INPUT"Delete what line";DEL:DELETE DEL:GOTO 310
W 400  IF K$="C" THEN GOSUB 1650:INPUT"Copy what line";FROM:INPUT"Copy to what line";TO:INPUT"How many lines";HOW:LIST FROM-HOW:DELETE FROM-HOW:LIST TO-1:DELETE TO-1:GOTO 310
G 410  IF K$="F" THEN GOSUB 1650:INPUT"Find what";FIND$:LIST X-1:GOTO 310
L 420  IF K$="N" THEN GOSUB 1650:INPUT"Find next";FIND$:LIST X-1:GOTO 310
N 430  IF K$="A" THEN GOSUB 1650:INPUT"Change what";FIND$:INPUT"Change to what";CHANGE$:LIST X-1:GOTO 310
B 440  IF K$="T" THEN GOSUB 1650:INPUT"Trace from what line";TRACE:LIST TRACE-1:GOTO 310
P 450  IF K$="P" THEN GOSUB 1650:INPUT"Print from what line";PRINTFROM:INPUT"Print to what line";PRINTTO:LPRINT"Program listing":LIST#1,PRINTFROM-PRINTTO:GOTO 310
PANY 460  IF K$="I" THEN GOSUB 1650:INPUT"Insert before what line";INSERT:LIST INSERT-1:GOTO 310
PANY 470  IF K$="E" THEN GOSUB 1650:INPUT"Edit what line";EDIT:LIST EDIT-1:GOTO 310
X 490  IF K$="U" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
```

```
T 500  IF K$="O" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
N 510  IF K$="K" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
UQ 520  IF K$="J" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
UQ 530  IF K$="X" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
A 540  IF K$="S" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
S 550  IF K$="V" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
UJ 580  IF K$="C" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
D 590  IF K$="V" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
J 600  IF K$="B" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
E 610  IF K$="N" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
W 620  IF K$="M" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
M 630  IF K$="," THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
E 640  IF K$="." THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
WU 650  IF K$="/" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
V 670  IF K$="I" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
S 680  IF K$="E" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
Y 690  IF K$="L" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
N 700  IF K$="A" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
X 710  IF K$="S" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
U 720  IF K$="T" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
G 730  IF K$="P" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
K 740  IF K$="Q" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
KZ 750  IF K$="R" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
Z 760  IF K$="W" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
Y 770  IF K$="Y" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
P 780  IF K$="Z" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
E 790  IF K$="V" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
EH 800  IF K$="B" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
GA 820  IF K$="N" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
P 840  IF K$="M" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
PH 850  IF K$="," THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
N 860  IF K$="." THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
KL 870  IF K$="/" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
KLE 880  IF K$="I" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
E 890  IF K$="E" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
```

```
N 510  IF K$="L" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
UQ 520  IF K$="J" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
UQ 530  IF K$="X" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
A 540  IF K$="S" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
S 550  IF K$="V" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
V 560  IF K$="C" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
U 570  IF K$="V" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
UJ 580  IF K$="B" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
D 590  IF K$="N" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
J 600  IF K$="M" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
E 610  IF K$="," THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
W 620  IF K$="." THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
M 630  IF K$="/" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
E 640  IF K$="I" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
WU 650  IF K$="E" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
V 670  IF K$="L" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
S 680  IF K$="A" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
Y 690  IF K$="S" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
N 700  IF K$="T" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
X 710  IF K$="P" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
U 720  IF K$="Q" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
G 730  IF K$="R" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
K 740  IF K$="W" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
KZ 750  IF K$="Y" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
Z 760  IF K$="Z" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
Y 770  IF K$="V" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
P 780  IF K$="B" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
E 790  IF K$="N" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
EH 800  IF K$="M" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
GA 820  IF K$="," THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
P 840  IF K$="." THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
PH 850  IF K$="/" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
N 860  IF K$="I" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
KL 870  IF K$="E" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
KLE 880  IF K$="L" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
E 890  IF K$="A" THEN GOSUB 1650:INPUT"Renumber starting at";START:INPUT"Renumber by";BY:RENUM START,,BY:GOTO 310
```

Continued

NANOEDITOR *Continued*

IBM PC/PCjr, TANDY 1000

```
900 '
910 '
920 '
930 '
940 '
950 '
960 '
970 '
980 '
990 '
1000 '
1010 '
1020 '
1030 '
1040 '
1050 '
1060 '
1070 '
1080 '
1090 '
1100 '
1110 '
1120 '
1130 '
1140 '
1150 '
1160 '
1170 '
1180 '
1190 '
1200 '
1210 '
1220 '
1230 '
1240 '
1250 '
1260 '
1270 '
1280 '
1290 '
1300 '
1310 '
1320 '
1330 '
1340 '
1350 '
```

```
1360 '
1370 '
1380 '
1390 '
1400 '
1410 '
1420 '
1430 '
1440 '
1450 '
1460 '
1470 '
1480 '
1490 '
1500 '
1510 '
1520 '
1530 '
1540 '
1550 '
1560 '
1570 '
1580 '
1590 '
1600 '
1610 '
1620 '
1630 '
1640 '
1650 '
1660 '
1670 '
1680 '
1690 '
1700 '
1710 '
1720 '
1730 '
1740 '
1750 '
1760 '
1770 '
1780 '
1790 '
1800 '
1810 '
1820 '
1830 '
1840 '
1850 '
1860 '
```

HCM

NANOASSEMBLER

TI-99/4A

```
100 '
110 '
120 '
130 '
140 '
150 '
160 '
170 '
180 '
190 '
200 '
210 '
220 '
230 '
```

```
240 '
250 '
260 '
270 '
280 '
290 '
300 '
310 '
320 '
330 '
340 '
350 '
360 '
370 '
380 '
390 '
```

Continued

```
L  400 IF (K<49)+(K>50)THEN 370
D  410 GOSUB 1430
W  420 IF FL$="" THEN 380
P  430 GOSUB 1700
S  440 GOTO 1970
N  450 STOP
C  460 REM ASSEMBLE
R  470 PRINT "EXECUTING FIRST PASS"
N  480 CA=0
K  490 LB=0
B  500 FOR LN=1 TO NL
J  510 LFLAG=0
V  520 GOSUB 1560
Y  530 IF SEG$(TL$,1,6)="      " THEN 750
U  540 IF LB=24 THEN 1230
L  550 LB$(LB)=SEG$(TL$,1,6)
P  560 X=POS(LB$(LB)," ",1)
D  570 IF X=0 THEN 640
B  580 FOR IT=6 TO X STEP -1
Z  590 IF SEG$(LB$(LB),IT,1)=" " THEN 610
F  600 GOTO 1210
I  610 LB$(LB)=SEG$(LB$(LB),1,IT-1)
U  620 NEXT IT
N  630 IF LB=0 THEN 670
C  640 FOR IT=0 TO LB-1
F  650 IF LB$(LB)=SEG$(LB$(IT),2,LEN(LB$(IT)))THEN 1250
K  660 NEXT IT
G  670 IF (ASC(LB$(LB))>64)*(ASC(LB$(LB))<91)THEN 690
Z  680 GOTO 1210
B  690 FOR IT=1 TO LEN(LB$(LB))
F  700 X=ASC(SEG$(LB$(LB),IT,1))
Q  710 IF ((X>64)*(X<92))+((X>47)*(X<58))THEN 730
E  720 GOTO 1210
Q  730 NEXT IT
H  740 LFLAG=1
R  750 IF SEG$(TL$,7,1)<>" " THEN 1190
P  760 GOSUB 1780
N  770 IF CA+NN(OP)>256 THEN 1310
A  780 ON OP+1 GOTO 1990,2210,2120,2120,1990,1990,1990,1990,1990,1990,1990,2120,2120,2120,2120,2120,2120,2020,2060,2260
X  790 IF LFLAG=0 THEN 820
N  800 LB$(LB)=CHR$(CA)&LB$(LB)
C  810 LB=LB+1
O  820 CA=CA+NN(OP)
W  830 NEXT LN
G  840 PRINT "EXECUTING SECOND PASS"
I  850 CA=0
G  860 FOR LN=1 TO NL
H  870 GOSUB 1560
E  880 GOSUB 1780
Z  890 EA=CA+NN(OP)
D  900 IF OP=17 THEN 1050
M  910 IF OP<>18 THEN 950
P  920 GOSUB 2420
B  930 EA=V
U  940 GOTO 1050
F  950 IF NN(OP)<3 THEN 1070
D  960 IF AD(CA+1)<>-1 THEN 1070
Y  970 FOR IT=0 TO LB
Y  980 IF SEG$(TL$,X+1,LL)=SEG$(LB$(IT),2,LEN(LB$(IT)))THEN 1010
A  990 NEXT IT
G 1000 GOTO 1290
R 1010 V=ASC(LB$(IT))
C 1020 AD(CA+1)=V-INT(V/16)*16
I 1030 AD(CA+2)=INT(V/16)
I 1040 GOTO 1070
A 1050 PRINT #PR:"        ";LN;LI$(LN)
I 1060 GOTO 1120
K 1070 PRINT #PR:"        ";LN;LI$(LN)
F 1080 FOR IT=0 TO NN(OP)-1
O 1090 GOSUB 1500
B 1100 PRINT #PR:OT$
F 1110 NEXT IT
R 1120 CA=EA
U 1130 NEXT LN
L 1140 PRINT "PRESS ANY KEY TO CONTINUE"
X 1150 GOSUB 2910
H 1160 IF PR=0 THEN 1180
L 1170 CLOSE #1
C 1180 RETURN
K 1190 PRINT #PR:"ILLEGAL CODE IN LINE";LN
Z 1200 GOTO 1330
W 1210 PRINT #PR:"ILLEGAL LABEL IN LINE";LN
R 1220 GOTO 1330
Z 1230 PRINT #PR:"TOO MANY LABELS--     25 MAXIMUM"
R 1240 GOTO 1330
P 1250 PRINT #PR:"DUPLICATE LABEL IN LINE";LN
Z 1260 GOTO 1330
P 1270 PRINT #PR:"NUMBER OUT OF RANGE IN LINE";LN
P 1280 GOTO 1330
M 1290 PRINT #PR:"ILLEGAL USE OF LABEL IN LINE";LN
K 1300 GOTO 1330
N 1310 PRINT #PR:"ADDRESS OUT OF RANGE IN LINE";LN
C 1320 GOTO 1330
T 1330 GOSUB 2950
Y 1340 GOTO 1970
D 1350 REM FILES
H 1360 ON K-48 GOTO 1370,1410
Z 1370 PRINT "INPUT FILE NAME FOR DSK1"
O 1380 INPUT FL$
S 1390 FL$="DSK1."&SEG$(FL$,1,8)&"_S"
T 1400 GOTO 1420
A 1410 FL$="CS1"
F 1420 RETURN
I 1430 ON K-48 GOTO 1440,1480
N 1440 PRINT "INPUT FILE NAME FOR DSK1"
K 1450 INPUT FL$
T 1460 FL$="DSK1."&SEG$(FL$,1,10)
N 1470 GOTO 1490
D 1480 FL$="CS1"
O 1490 RETURN
I 1500 OT$=STR$(CA+IT)
A 1510 OT$=SEG$("000",1,3-LEN(OT$))&OT$
R 1520 OU$=STR$(AD(CA+IT))
T 1530 OU$=SEG$("000",1,3-LEN(OU$))&OU$
T 1540 OT$=OT$&"  "&OU$
N 1550 RETURN
A 1560 TL$=SEG$(LI$(LN),1,24)
O 1570 FOR IT=LEN(TL$)TO 1 STEP -1
I 1580 IF SEG$(TL$,IT,1)<>" " THEN 1610
L 1590 TL$=SEG$(TL$,1,LEN(TL$)-1)
H 1600 NEXT IT
U 1610 LL=IT
N 1620 RETURN
D 1630 OPEN #1:FL$,INTERNAL,INPUT ,FIXED 64
B 1640 INPUT #1:NL
V 1650 FOR IT=1 TO NL
C 1660 INPUT #1:LI$(IT)
G 1670 NEXT IT
Z 1680 CLOSE #1
W 1690 RETURN
G 1700 OPEN #1:FL$,INTERNAL,OUTPUT,FIXED 64
A 1710 PRINT "SAVING FILE. . ."
Q 1720 FOR IT=0 TO 245 STEP 7
K 1730 PRINT #1:AD(IT);AD(IT+1);AD(IT+2);AD(IT+3);AD(IT+4);AD(IT+5);AD(IT+6)
A 1740 NEXT IT
N 1750 PRINT #1:AD(252);AD(253);AD(254);AD(255)
A 1760 CLOSE #1
D 1770 RETURN
B 1780 FOR IT=0 TO 18
Z 1790 IF SEG$(TL$,8,LEN(OP$(IT)))=OP$(IT)THEN 1820
N 1800 NEXT IT
M 1810 GOTO 1190
S 1820 OP=IT
Z 1830 X=8+LEN(OP$(OP))
K 1840 IF (OP=9)*(SEG$(TL$,X,1)="G")THEN 1800
Q 1850 RETURN
S 1860 PRINT "SEND ASSEMBLY TO:": :"1. SCREEN":"2. PRINTER"
W 1870 GOSUB 2860
H 1880 IF K=50 THEN 1910
V 1890 PR=0
S 1900 RETURN
A 1910 PRINT "ENTER PRINTER DEVICE :"
K 1920 INPUT PR$
V 1930 PR=1
Z 1940 OPEN #PR:PR$
N 1950 RETURN
M 1960 REM END PROGRAM
U 1970 PRINT
P 1980 END
K 1990 IF X<>11 THEN 1190
E 2000 AD(CA)=OP
A 2010 GOTO 790
O 2020 GOSUB 2420
O 2030 IF V=-1 THEN 1290
T 2040 AD(CA)=V
G 2050 GOTO 790
W 2060 GOSUB 2420
O 2070 IF V=-1 THEN 1290
U 2080 IF LFLAG=0 THEN 1190
A 2090 LB$(LB)=CHR$(V)&LB$(LB)
I 2100 LB=LB+1
A 2110 GOTO 830
N 2120 GOSUB 2420
I 2130 AD(CA)=OP
O 2140 IF V<>-1 THEN 2180
T 2150 AD(CA+1)=-1
V 2160 AD(CA+2)=-1
D 2170 GOTO 790
U 2180 AD(CA+1)=V-INT(V/16)*16
Q 2190 AD(CA+2)=INT(V/16)
L 2200 GOTO 790
N 2210 GOSUB 2420
F 2220 IF V=-1 THEN 1290
T 2230 AD(CA)=OP
V 2240 AD(CA+1)=V
U 2250 GOTO 790
S 2260 GOSUB 2420
T 2270 IF (V=-1)+(LFLAG=1)THEN 1290
X 2280 CA=V
N 2290 GOTO 830
Y 2300 V=0
W 2310 FOR IT=0 TO LEN(T$)-1
Y 2320 V=V+VAL(SEG$(T$,IT+1,1))*2^(LEN(T$)-IT-1)
E 2330 NEXT IT
F 2340 IF (((OP=1)+(OP=16))*(V>15))+(V>255)THEN 1270
G 2350 RETURN
```

Continued

TYPE-IN LISTINGS

NANOASSEMBLER *Continued*

TI-99/4A

```
K 2360 V=0
Q 2370 FOR IT=0 TO LEN(T$)-1
O 2380 V=V+16^(LEN(T$)-IT-1)*(POS(HX$,SEG$
       (T$,IT+1,1),1)-1)
C 2390 NEXT IT
G 2400 IF (((OP=1)+(OP=16))*(V>15))+(V>255
       )THEN 1270
F 2410 RETURN
B 2420 IF SEG$(TL$,X,1)<>" " THEN 1270
A 2430 IF SEG$(TL$,X+1,1)="" THEN 1190
A 2440 Y=POS(NM$,SEG$(TL$,X+1,1),1)
K 2450 IF Y<>0 THEN 2480
F 2460 V=-1
D 2470 RETURN
F 2480 IF Y<=2 THEN 2500
Y 2490 Y=3
A 2500 ON Y GOTO 2510,2580,2650
N 2510 T$=""
A 2520 FOR IT=X+2 TO LL
Q 2530 N$=(SEG$(TL$,IT,1))
Z 2540 IF (N$<"0")+(N$>"1")THEN 1270
N 2550 T$=T$&N$
T 2560 NEXT IT
A 2570 GOTO 2300
K 2580 T$=""
R 2590 FOR IT=X+2 TO LL
Y 2600 N$=(SEG$(TL$,IT,1))
G 2610 IF POS(HX$,N$,1)=0 THEN 1270
V 2620 T$=T$&N$
L 2630 NEXT IT
G 2640 GOTO 2360
Q 2650 T$=""
M 2660 FOR IT=X+1 TO LL
X 2670 N$=(SEG$(TL$,IT,1))
A 2680 IF (N$<"0")+(N$>"9")THEN 1270
K 2690 T$=T$&N$
N 2700 NEXT IT
N 2710 V=VAL(T$)
N 2720 RETURN
E 2730 REM PRINT MENU
S 2740 CALL CLEAR
P 2750 READ NI
B 2760 FOR IT=1 TO NI
K 2770 READ A,A$
F 2780 PRINT :TAB(A);A$
Y 2790 NEXT IT
L 2800 PRINT : : : : :
N 2810 RETURN
E 2820 DATA 7,5,The NanoAssembler,1," ",1,
       LOAD SOURCE FILE FROM:
L 2830 DATA 8,(1) DISK (DSK1),8,(2) CASSET
       TE (CS1),1," ",5,YOUR CHOICE:
K 2840 DATA 7,5,The NanoAssembler,1," ",7,
       SAVE OBJECT FILE TO:
N 2850 DATA 8,(1) DISK (DSK1),8,(2) CASSET
       TE (CS1),1," ",5,YOUR CHOICE:
A 2860 CALL KEY(0,K,S)
N 2870 IF S=0 THEN 2860
E 2880 IF (K<49)+(K>50)THEN 2870
R 2890 PRINT CHR$(K)
N 2900 RETURN
I 2910 CALL KEY(0,K,S)
R 2920 IF S=0 THEN 2910
R 2930 CALL HCHAR(18,20,K)
N 2940 RETURN
E 2950 FOR IT=1 TO 1000
F 2960 NEXT IT
I 2970 RETURN
R 2980 REM PROGRAM INIT
X 2990 RESTORE 3060
K 3000 FOR IT=0 TO 18
J 3010 READ OP$(IT),NN(IT)
P 3020 NEXT IT
F 3030 HX$="0123456789ABCDEF"
J 3040 NM$="%$0123456789"
G 3050 RETURN
E 3060 DATA ADD,1,LDA#,2,LDA,3,STA,3,TAB,1
       ,TBA,1,RRC,1,RLC,1,AND,1,OR,1,XOR,1
A 3070 DATA BZ,3,BNZ,3,BCS,3,BCC,3,JMP,3,D
       N,1,EQU,0,ORG,0
```

HCM

NANOEDITOR

TI-99/4A

```
P 100 REM **************
Q 110 REM * NANOEDITOR *
P 120 REM **************
M 130 REM COPYRIGHT 1985
H 140 REM EMERALD VALLEY PUBLISHING CO
A 150 REM BY ROGER WOOD
R 160 REM HOME COMPUTER MAGAZINE
G 170 REM VERSION 5.6.1
T 180 REM TI BASIC AND EXTENDED BASIC
M 190 REM
F 200 CALL CLEAR
D 210 PRINT TAB(9);"The NanoEditor": : :
      :TAB(6);"PLACE ALPHA LOCK DOWN": :"
        PLEASE WAIT WHILE I SET UP"
Q 220 DIM LI$(200)
T 230 GOSUB 3450
A 240 RESTORE 3320
V 250 GOSUB 3120
J 260 GOSUB 3370
Y 270 IF (K<49)+(K>52)THEN 260
E 280 ON K-48 GOSUB 310,1590,2030,2160
W 290 GOTO 240
D 300 REM EDIT
D 310 CALL CLEAR
A 320 LN$="CMD"
I 330 L=1
Y 340 RW=1
D 350 B=ASC(SEG$(CMD$,1,1))
V 360 T=ASC(SEG$(CMD$,LEN(CMD$),1))
R 370 CN$=SEG$(C$,1,3)
Z 380 S$=""
C 390 E=0
B 400 GOSUB 2250
W 410 IF ESC=0 THEN 430
S 420 RETURN
K 430 OP=POS(CMD$,S$,1)
Q 440 IF (S$="")+(OP=0)THEN 380
H 450 IF OP=1 THEN 610
M 460 CN$=SEG$(C$,1,6)
S 470 LN$="L#?"
A 480 L=3
R 490 RW=2
S 500 B=ASC("0")
Y 510 T=ASC("9")
H 520 S$=""
Z 530 GOSUB 2250
E 540 IF ESC=0 THEN 560
S 550 RETURN
U 560 IF LEN(S$)=0 THEN 520
M 570 LN=VAL(S$)
I 580 IF (LN>0)*(LN<201)THEN 600
W 590 GOTO 520
E 600 LN$=SEG$("000",1,3-LEN(S$))&S$
A 610 ON OP GOSUB 640,850,970,1160,1360
B 620 GOTO 310
X 630 REM ADD
D 640 IF NL=200 THEN 790
W 650 LN=NL+1
Q 660 GOSUB 3230
B 670 FOR RW=3 TO 21 STEP 2
N 680 GOSUB 3200
B 690 S$=""
A 700 GOSUB 2250
M 710 IF ESC=0 THEN 730
J 720 RETURN
H 730 IF LEN(S$)=0 THEN 690
P 740 LI$(LN)=S$
D 750 NL=LN
A 760 LN=LN+1
Y 770 IF NL=200 THEN 810
N 780 NEXT RW
B 790 CALL CLEAR
I 800 GOTO 670
G 810 PRINT "SOURCE CODE IS A MAXIMUM OF
      200 LINES!"
Y 820 GOSUB 3410
F 830 RETURN
A 840 REM DELETE
X 850 IF LN<=NL THEN 890
F 860 PRINT "NO SUCH LINE"
P 870 GOSUB 3410
C 880 RETURN
D 890 PRINT "DELETING LINE";LN
H 900 FOR IT=LN TO NL
R 910 LI$(IT)=LI$(IT+1)
T 920 NEXT IT
J 930 NL=NL-1
S 940 LI$(NL+1)=""
S 950 RETURN
R 960 REM EDIT
P 970 IF LN<=NL THEN 1010
K 980 PRINT "THERE IS NO LINE ";LN$
E 990 GOSUB 3410
Z 1000 RETURN
I 1010 GOSUB 3230
L 1020 FOR RW=3 TO 21 STEP 2
V 1030 GOSUB 3200
F 1040 S$=LI$(LN)
F 1050 GOSUB 2250
Y 1060 IF ESC=0 THEN 1080
G 1070 RETURN
W 1080 IF LEN(S$)=0 THEN 890
K 1090 LI$(LN)=S$
N 1100 IF NL=LN THEN 950
S 1110 LN=LN+1
I 1120 NEXT RW
A 1130 CALL CLEAR
E 1140 GOTO 1020
P 1150 REM INSERT
C 1160 IF (LN<=NL)*(NL<200)THEN 1200
O 1170 PRINT "LAST LINE IS";NL;"CAN'T INSE
       RT"
C 1180 GOSUB 3410
N 1190 RETURN
A 1200 RW=3
W 1210 GOSUB 3230
C 1220 GOSUB 3200
T 1230 GOSUB 3280
X 1240 S$=""
P 1250 GOSUB 2250
U 1260 IF ESC=0 THEN 1280
E 1270 RETURN
T 1280 IF LEN(S$)=0 THEN 1250
M 1290 FOR IT=NL TO LN STEP -1
N 1300 LI$(IT+1)=LI$(IT)
R 1310 NEXT IT
```

Continued

TYPE-IN LISTINGS

NANOEDITOR *Continued*

```
100 CALL CLEAR :: CALL SCREEN(2) :: CALL COLOR(13,2,1) :: CALL COLOR(14,7,1)
110 DISPLAY AT(1,1)ERASE ALL:"NANOEDITOR" :: DISPLAY AT(3,1):"Type in the fi
le name you wish to edit." :: DISPLAY AT(5,1):"Example: DSK1.FILENAME" :: DI
SPLAY AT(7,1):"Name? " :: ACCEPT AT(7,7):F$ :: IF F$="" THEN 110
120 OPEN #1:F$,VARIABLE 163,INPUT :: DISPLAY AT(9,1):"Loading file..." :: I
F EOF(1)=1 THEN 150
130 LINPUT #1:A$(I) :: I=I+1 :: GOTO 120
140 ! ** Edit Mode **
150 CLOSE #1 :: DISPLAY AT(9,1):"Press any key to continue" :: CALL KEY(0,K
,S) :: IF S=0 THEN 150
160 CALL CLEAR :: DISPLAY AT(1,1):"Edit Mode" :: DISPLAY AT(3,1):"Enter Lin
e Number" :: ACCEPT AT(3,17):L :: IF L<1 OR L>I-1 THEN 160
170 DISPLAY AT(5,1):"Edit Line:" :: DISPLAY AT(6,1):A$(L) :: DISPLAY AT(8,1
):"Press any key to continue" :: CALL KEY(0,K,S) :: IF S=0 THEN 170
180 DISPLAY AT(10,1):"Enter Edit Command" :: ACCEPT AT(10,20):C$ :: IF C$="
" THEN 180
190 IF C$="I" THEN 200 ELSE IF C$="D" THEN 210 ELSE IF C$="R" THEN 220 ELSE
 IF C$="S" THEN 230 ELSE IF C$="E" THEN 240 ELSE 180
200 DISPLAY AT(12,1):"Insert Text:" :: ACCEPT AT(13,1):T$ :: A$(L)=A$(L)&T$
 :: GOTO 160
210 A$(L)="" :: GOTO 160
220 DISPLAY AT(12,1):"Replace Text:" :: ACCEPT AT(13,1):T$ :: A$(L)=T$ :: G
OTO 160
230 DISPLAY AT(12,1):"Search For:" :: ACCEPT AT(13,1):S$ :: P=POS(A$(L),S$,
1) :: IF P=0 THEN 160
240 DISPLAY AT(15,1):"Replace With:" :: ACCEPT AT(16,1):R$ :: A$(L)=SEG$(A$
(L),1,P-1)&R$&SEG$(A$(L),P+LEN(S$),LEN(A$(L))) :: GOTO 160
250 ! ** Save File **
260 DISPLAY AT(1,1)ERASE ALL:"Save File" :: DISPLAY AT(3,1):"Enter file name
 to save as:" :: ACCEPT AT(4,1):F$ :: IF F$="" THEN 260
270 OPEN #1:F$,VARIABLE 163,OUTPUT :: FOR J=1 TO I-1 :: PRINT #1:A$(J) :: N
EXT J :: CLOSE #1
280 DISPLAY AT(6,1):"File saved!" :: DISPLAY AT(8,1):"Press any key to conti
nue" :: CALL KEY(0,K,S) :: IF S=0 THEN 280
290 ! ** Quit **
300 DISPLAY AT(1,1)ERASE ALL:"Quit Editor" :: DISPLAY AT(3,1):"Are you sure?
 (Y/N)" :: ACCEPT AT(3,16):Q$ :: IF Q$="Y" THEN 310 ELSE IF Q$="N" THEN 160
 ELSE 300
310 DISPLAY AT(5,1):"Goodbye!" :: END
```

Continued

TYPE-IN LISTINGS

NANOEDITOR *Continued*

TI-99/4A

```
U 3350 DATA 6,7,LOAD / SAVE MENU,10,(1) SA
       VE,10,(2) LOAD,10,(3) EXIT,1," ",5,
       "YOUR CHOICE:
W 3360 DATA 4,10,PRINT FILE,1," ",3,ENTER
       PRINTER DEVICE,3,OR PRESS [ENTER] T
       O EXIT,
P 3370 CALL KEY(0,K,S)
D 3380 IF S=0 THEN 3370
M 3390 CALL HCHAR(18,20,K)
F 3400 RETURN
U 3410 FOR IT=1 TO 1000
N 3420 NEXT IT
I 3430 RETURN
X 3440 REM PROGRAM INIT
M 3450 C$=CHR$(13)&CHR$(15)&CHR$(3)&CHR$(9
       )&CHR$(8)&CHR$(7)&CHR$(1)
L 3460 CMD$="ADEIL"
O 3470 RETURN
```

HCM

BASID'S BASIC LOADER

COMMODORE 64

```
H  100 REM ***********************
M  110 REM * BASID'S BASIC LOADER *
T  120 REM ***********************
M  130 REM COPYRIGHT 1985
R  140 REM EMERALD VALLEY PUBLISHING CO.
F  150 REM BY RANDY THOMPSON
N  160 REM HOME COMPUTER MAGAZINE
A  170 REM VERSION 5.6.1
P  180 REM C-64 BASIC
T  190 PRINT"[SHIFT CLR]INITIALIZING..."
M  200 FOR I=49152 TO 50461:READ D:C=C+D:P
       OKE I,D:NEXT
N  210 IF C<>157106 THEN PRINT"** ERROR IN
        DATA STATEMENTS **":END
J  220 SYS 49152:PRINT"** BA[CTRL RVS ON]SI
       D[CTRL RVS OFF] ACTIVATED **"
M  230 DATA 169,51,141,4,3,169,192,141
J  240 DATA 5,3,169,246,141,6,3,169
S  250 DATA 192,141,7,3,169,43,141,8
Y  260 DATA 3,169,193,141,9,3,169,82
F  270 DATA 141,10,3,169,193,141,11,3
O  280 DATA 169,132,141,0,3,169,193,141
V  290 DATA 1,3,96,166,122,160,4,132
G  300 DATA 15,189,0,2,16,7,201,255
P  310 DATA 240,62,232,208,244,201,32,240
Q  320 DATA 55,133,8,201,34,240,85,36
H  330 DATA 15,112,45,201,63,208,4,169
Z  340 DATA 153,208,37,201,48,144,4,201
S  350 DATA 60,144,29,132,113,160,0,132
P  360 DATA 11,136,134,122,202,200,232,189
E  370 DATA 0,2,56,249,158,160,240,245
L  380 DATA 201,128,208,47,5,11,164,113
M  390 DATA 232,200,153,251,1,201,0,240
O  400 DATA 56,56,233,58,240,4,201,73
E  410 DATA 208,2,133,15,56,233,85,208
I  420 DATA 160,133,8,189,0,2,240,224
A  430 DATA 197,8,240,220,200,153,251,1
E  440 DATA 232,208,240,166,122,230,11,200
Q  450 DATA 185,157,160,16,250,185,158,160
L  460 DATA 208,181,240,15,189,0,2,16
R  470 DATA 189,153,253,1,198,123,169,255
Y  480 DATA 133,122,96,160,0,185,153,193
Z  490 DATA 208,2,200,232,189,0,2,56
L  500 DATA 249,153,193,240,245,201,128,20
       8
J  510 DATA 4,5,11,208,153,166,122,230
Q  520 DATA 11,200,185,152,193,16,250,185
P  530 DATA 153,193,208,224,240,198,16,15
W  540 DATA 36,15,48,11,201,255,240,7
I  550 DATA 201,204,176,6,76,36,167,76
R  560 DATA 243,166,56,233,203,170,132,73
E  570 DATA 160,255,202,240,8,200,185,153
O  580 DATA 193,16,250,48,245,200,185,153
Z  590 DATA 193,48,5,32,71,171,208,245
K  600 DATA 76,239,166,32,115,0,32,52
F  610 DATA 193,76,174,167,201,204,144,4
J  620 DATA 201,223,144,6,32,121,0,76
R  630 DATA 237,167,56,233,204,10,170,189
C  640 DATA 226,193,72,189,225,193,72,76
B  650 DATA 115,0,169,0,133,13,32,115
L  660 DATA 0,201,204,144,4,201,221,144
G  670 DATA 6,32,121,0,76,141,174,56
Z  680 DATA 233,204,10,72,32,115,0,32
A  690 DATA 241,174,104,168,185,7,194,133
C  700 DATA 85,185,8,194,133,86,32,84
R  710 DATA 0,76,141,173,224,11,208,14
Y  720 DATA 201,76,144,10,201,95,176,6
P  730 DATA 56,233,76,76,69,193,76,139
H  740 DATA 227,86,79,204,70,82,209,87
B  750 DATA 65,86,197,65,84,84,65,67
V  760 DATA 203,68,69,67,65,217,83,85
G  770 DATA 211,82,69,204,83,79,85,78
O  780 DATA 196,70,73,76,84,69,210,76
K  790 DATA 207,66,65,206,72,201,82,73
P  800 DATA 78,199,83,89,78,195,82,69
Z  810 DATA 211,67,85,212,80,85,76,83
K  820 DATA 197,76,83,212,73,78,73,212
T  830 DATA 0,170,194,195,194,209,194,13
Z  840 DATA 195,38,195,250,194,19,195,44
H  850 DATA 195,65,195,106,195,110,195,114
N  860 DATA 195,149,195,170,195,191,195,21
       7
A  870 DATA 195,252,195,18,196,46,196,58
P  880 DATA 196,67,196,86,196,115,196,128
J  890 DATA 196,121,196,140,196,152,196,16
       4
I  900 DATA 196,184,196,188,196,192,196,20
       5
W  910 DATA 196,221,196,237,196,243,196,11
G  920 DATA 197,157,0,212,157,145,194,96
W  930 DATA 32,235,183,76,66,194,32,247
H  940 DATA 183,201,0,240,3,76,72,178
K  950 DATA 166,20,138,202,240,14,202,240
Q  960 DATA 9,202,240,3,76,72,178,162
O  970 DATA 14,96,162,7,96,32,48,194
W  980 DATA 165,21,208,240,165,20,201,16
N  990 DATA 176,234,133,2,96,32,85,194
N 1000 DATA 10,10,10,10,133,2,96,189
U 1010 DATA 150,194,74,74,74,74,168,76
H 1020 DATA 162,179,32,48,194,165,21,240
E 1030 DATA 3,76,72,178,165,20,201,2
Z 1040 DATA 176,247,232,232,232,232,201,0
L 1050 DATA 96,0,0,0,0,0,0,0
F 1060 DATA 0,0,0,0,0,0,0,0
E 1070 DATA 0,0,0,0,0,0,0,0
J 1080 DATA 0,0,0,32,158,183,224,16
B 1090 DATA 144,3,76,72,178,173,169,194
C 1100 DATA 41,240,133,2,138,5,2,162
K 1110 DATA 24,76,41,194,32,48,194,165
X 1120 DATA 20,32,41,194,232,165,21,76
S 1130 DATA 41,194,32,48,194,165,21,240
I 1140 DATA 3,76,72,178,165,20,240,249
I 1150 DATA 201,5,176,245,168,169,8,10
T 1160 DATA 136,208,252,133,2,232,232,232
C 1170 DATA 232,189,145,194,41,15,5,2
J 1180 DATA 76,41,194,32,101,194,232,232
U 1190 DATA 232,232,232,232,189,145,194,41
T 1200 DATA 15,5,2,76,41,194,32,101
A 1210 DATA 194,76,255,194,32,85,194,232
A 1220 DATA 232,232,232,232,232,189,145,19
       4
R 1230 DATA 41,240,5,2,76,41,194,32
A 1240 DATA 85,194,76,24,195,32,122,194
H 1250 DATA 240,8,189,145,194,9,1,76
R 1260 DATA 41,194,189,145,194,41,254,76
U 1270 DATA 41,194,32,48,194,201,3,208
Z 1280 DATA 2,169,4,166,21,240,3,76
J 1290 DATA 72,178,166,20,224,2,176,247
S 1300 DATA 224,0,240,5,13,168,194,208
U 1310 DATA 5,73,255,45,168,194,162,23
F 1320 DATA 76,41,194,169,16,208,6,169
Q 1330 DATA 32,208,2,169,64,133,2,32
O 1340 DATA 158,183,224,2,144,3,76,72
O 1350 DATA 178,165,2,224,0,240,5,13
Q 1360 DATA 169,194,208,5,73,255,45,169
S 1370 DATA 194,162,24,76,41,194,32,122
T 1380 DATA 194,240,8,189,145,194,9,4
Q 1390 DATA 76,41,194,189,145,194,41,251
N 1400 DATA 76,41,194,32,122,194,240,8
F 1410 DATA 189,145,194,9,2,76,41,194
A 1420 DATA 189,145,194,41,253,76,41,194
L 1430 DATA 32,158,183,224,16,144,3,76
S 1440 DATA 72,178,138,32,104,194,173,168
X 1450 DATA 194,41,15,5,2,162,23,76
S 1460 DATA 41,194,32,138,173,32,247,183
L 1470 DATA 201,8,144,3,76,72,178,165
P 1480 DATA 20,162,21,32,41,194,160,3
N 1490 DATA 70,21,102,20,136,208,249,232
P 1500 DATA 165,20,76,41,194,32,48,194
C 1510 DATA 165,21,201,16,176,222,232,232
K 1520 DATA 165,20,32,41,194,232,165,21
H 1530 DATA 76,41,194,169,26,160,196,76
D 1540 DATA 30,171,42,42,32,82,65,78
U 1550 DATA 68,89,32,84,72,79,77,80
J 1560 DATA 83,79,78,32,42,42,0,169
F 1570 DATA 0,162,24,32,41,194,202,16
Z 1580 DATA 250,96,173,169,194,41,15,168
K 1590 DATA 76,162,179,32,54,194,189,145
N 1600 DATA 194,133,99,189,146,194,133,98
R 1610 DATA 162,144,56,76,73,188,32,54
H 1620 DATA 194,160,0,189,149,194,41,240
M 1630 DATA 240,14,133,2,169,8,160,1
Q 1640 DATA 10,36,2,208,3,200,208,248
F 1650 DATA 76,162,179,32,54,194,76,111
Y 1660 DATA 194,32,54,194,232,76,111,194
P 1670 DATA 32,54,194,189,150,194,41,15
H 1680 DATA 168,76,162,179,32,54,194,189
I 1690 DATA 151,194,41,15,168,76,162,179
A 1700 DATA 32,54,194,189,149,194,41,1
K 1710 DATA 168,76,162,179,32,54,194,201
B 1720 DATA 3,208,2,169,4,45,168,194
I 1730 DATA 240,2,169,1,168,76,162,179
A 1740 DATA 169,16,208,6,169,32,208,2
O 1750 DATA 169,64,45,169,194,240,2,169
X 1760 DATA 1,168,76,162,179,32,54,194
A 1770 DATA 189,149,194,41,4,240,2,169
J 1780 DATA 1,168,76,162,179,32,54,194
C 1790 DATA 189,149,194,41,2,240,2,169
W 1800 DATA 1,168,76,162,179,173,168,194
M 1810 DATA 76,114,194,173,166,194,133,99
Q 1820 DATA 169,0,160,3,14,167,194,42
A 1830 DATA 136,208,249,133,98,162,144,56
P 1840 DATA 76,73,188,32,54,194,189,147
K 1850 DATA 194,133,99,189,148,194,133,98
O 1860 DATA 162,144,56,76,73,188
```

HCM

TYPE-IN LISTINGS

Subscription & Product Order Form

Enclose payment or credit card information & mail with completed form to:
Emerald Valley Publishing Co. ● P.O. Box 70288 ● Eugene, OR 97401
Or use our TOLL-FREE Order Line for VISA/MasterCard orders only:
1-800-828-2212 (Minimum $10. Order) In Oregon, Alaska, Hawaii Tel. (503) 485-8796
HOURS: MONDAY THROUGH FRIDAY 7:00AM TO 4:00PM; SATURDAY 9:00AM TO 1:00PM P.S.T.

For more information see Shopping List on inside page.
Defective media gladly exchanged. NO REFUNDS on media.
Offer & Prices Subject To Change Without Notice.

☐ Check or Money Order Enclosed **(MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK)**

Charge my ☐ VISA ☐ MasterCard Date Expires _____ Tel. No. _____

Account No. _____ Signature _____

PURCHASER — SHIPPING ADDRESS A *Please Print*

Name _____
Address _____
TO INSURE FASTER DELIVERY, STREET ADDRESS IS ESSENTIAL
City _____ State _____ Zip _____

SHIPPING ADDRESS B

Name _____
Address _____
TO INSURE FASTER DELIVERY, STREET ADDRESS IS ESSENTIAL
City _____ State _____ Zip _____

MAGAZINE & PROGRAM SUBSCRIPTIONS: (FOR OFFICE USE ONLY)
New Mag.: _____ CODE Old Mag.: _____ CODE New Program: _____ CODE Old Program: _____ CODE

If any portion of your order is to be shipped to address "B," please indicate your preference by checking box A or B for each item.

1 ON DISK or ON TAPE
(CURRENT ISSUE)
$4.95 + $1.00 ship. & handl.
$6.95 Canada — $10.95 Foreign Airmail
(Non-purchaser, non-subscriber price: $12.95)

(Please indicate your type of computer media)

DISK VERSION						CASSETTE VERSION		
APPLE	ATARI	C-64	IBM PC	IBM PCjr	TI	ATARI	C-64	TI

Volume	Issue	Price good Through
5	6	Dec. 31, 1985

AMOUNT $ — SHIP TO: A ADDRESS B

2 HOME COMPUTER MAGAZINE SUBSCRIPTION: NEW ☐ RENEWAL ☐ GIFT ☐

LABEL NO. H _____

	USA Surface	CANANDA Surface	FOREIGN Surface
1-YEAR (10 issue)	$25	$32	$46
2-YEAR (20 issue)	$45	$59	N/A
3-YEAR (30 issue)	$63	$84	N/A

(Please indicate type of computer media for your 2 premium issues)

DISK VERSION						CASSETTE VERSION		
APPLE	ATARI	C-64	IBM PC	IBM PCjr	TI	ATARI	C-64	TI

Office Use Only
Sub.:
Media:

Your Subscription will begin with the up-coming new issue. ☐ Please send a raincheck for the free software.

$ — SHIP TO: A ADDRESS B

3 HCM'S ON DISK or ON TAPE PROGRAM SUBSCRIPTION NEW ☐ RENEWAL ☐ GIFT ☐

(Please mark your choice)

	USA	CANADA
8 ISSUES	$44.95	$49.95
10 ISSUES	$49.95	$54.95

(Please indicate your type of computer media) LABEL NO. 8 _____

DISK VERSION						CASSETTE VERSION		
APPLE	ATARI	C-64	IBM PC	IBM PCjr	TI	ATARI	C-64	TI

$ — SHIP TO: A ADDRESS B

ADDITIONAL SPACE FOR ORDERING MERCHANDISE (See Shopping List On Inside)

ITEM NO.	QTY	Description	UNIT PRICE (Each)	SHIP. & Handl. (Each)	SHIPPING INSTRUCTIONS	
M E 0 1 0 1 0 6	1	**Music & Electronics Magazine** (6-Issue Charter Subscription) Includes 6 Magazines and 6 M & E ON TAPE™ Instructional Cassette Tapes (U.S. Newsstand Value $56.70)	$32.97 (U.S) $39.97 (CANADA)	INCLUDED	A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$

BACK ISSUES: HOME COMPUTER MAGAZINE and MEDIA*

(Please mark your choice)

(Please indicate your type of computer media)

DISK VERSION						CASSETTE VERSION		
APPLE	ATARI	C-64	IBM PC	IBM PCjr	TI	ATARI	C-64	TI

ISSUE	4-1	4-2	AUG 4-3	4-4	4-5	5-1	5-2	5-3	5-4	U.S. Surface	CANADA	FOREIGN Surface	FOREIGN Air	QTY.
Magazine										3.95	5.95	5.50	7.50	× =
Media										6.95	7.95	N/A	10.95	× =
Magazine/Media Set										9.90	12.95	13.95	17.50	× =

$ — SHIP TO: A ADDRESS B

BACK ISSUES: 99'er HOME COMPUTER MAGAZINE and MEDIA (COVERS ONLY TI COMPUTERS—THE FORERUNNER OF HCM)

(Please mark your choice) ☐ TAPE ☐ DISK (Minimum Order of 3)

ISSUE	1-6	NOV 2-1	DEC 2-2	JAN 2-3	FEB 2-4	MAR 2-5	APR 2-6	MAY 2-7	JUN 2-8	JUL 2-9	AUG 2-10	SEP 2-11	OCT 2-12	NOV 2-13
Magazine														
Media														
Magazine/Media Set														

QTY	MAGAZINE PRICES* U.S.	Canada	Foreign‡	MAG/MEDIA SET PRICES U.S.	Canada	Foreign‡	MEDIA PRICES U.S.	Canada	Foreign‡
3	$7.95	$10.95	$10.95	$15.95	$19.95	$21.95	$10.95	$12.95	$14.95
6	$12.95	$18.95	$18.95	$29.50	$37.95	$39.95	$20.95	$22.95	$25.95
12	$21.95	$29.95	$29.95	$49.95	$59.95	$63.95	$39.95	$44.95	$47.95

$ — SHIP TO: A ADDRESS B

FOR OFFICE USE ONLY

A	B		A	B	
☐	☐	Air Mail	☐	☐	2nd Class
☐	☐	First Class	☐	☐	3rd Class
☐	☐	UPS (Standard)	☐	☐	Air Canada
☐	☐	UPS (Blue Label)	☐	☐	4th Class
☐	☐	Other _____	☐	☐	Foreign Surface

NOTES:
‡ Magazines Shipped to Foreign Countries via Surface
* Atari coverage commenced with Volume 5, No. 5.
- Merchandise shipped via UPS—requires street address.
- Satisfaction Guaranteed—or the unfilled portion of your subscription will be refunded, less the cost of any premiums you have received.
- Machines covered: Apple II family, Atari 800/800-XL/130-XE, Commodore 64/128, IBM PC/PCjr, TI-99/4A.
- For subscriptions, please allow 6-8 weeks for your first issue —magazines and media shipped separately.

APO/FPO Include $3.00 Extra Per Total Merchandise Order $

SUBTOTAL $

Priority Merchandise Handling ADD $2.00 (SUBSCRIPTIONS NOT APPLICABLE) $

TOTAL $

ORDER "A" DATE SHIPPED: _____ BY: _____
ORDER "B" DATE SHIPPED: _____ BY: _____
BACK ORDER DATE SHIPPED: _____ BY: _____

COMMENTS: _____
ACCT: _____
DATE: _____

SPO2561185

For Apple II Family, C-64, IBM PC & PCjr Users

ON DISK Revue™ Volume 1

- 10 Complete Programs On A Ready-To-Run Disk
- Exciting Activities For The Entire Family—At An Affordable Price
- Entertainment, Productivity & Education Plus Valuable Programming Secrets

Valuable Proof-of-Purchase Coupon For Gift Redemption



An Attractive Library-Storage Slipcase

CONTENTS

1. Boolean Brain
2. Electronic Home Secretary
3. Savings Planner
4. Snap-Calc
5. Tablut
6. Wild Kingdom
7. Musical Mystery Words
8. Market Madness
9. Tower of Hanoi
10. Missile Math

64-page Full-Color Book



Disk Stored in Re-usable Protective Pouch

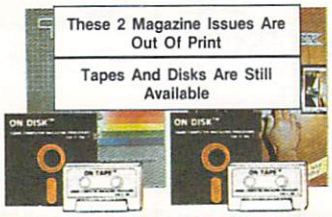
ONLY $14.95 plus $2.00 SHIPPING & HANDLING

AS LOW AS $1.83 EACH MAGAZINE!
See Order Card on Other side of this page
ABOUT $4 PER SET!

A Back-Issue/Software Bonanza of 99'er HOME COMPUTER magazine™ At Unbeatable Prices

The original 99'er Magazine and 99'er Home Computer Magazine were the forerunners of the present-day Home Computer Magazine. Each of these magazine back issues—exclusively covering the Texas Instruments TI-99/4A—is now available with your choice of either a floppy disk or a cassette tape that contains all the programs in that issue.

For Magazine Contents See "Shopping List" On Opposite Page







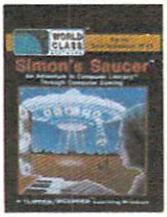


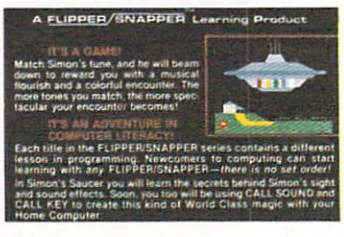
12 MAGAZINE & MEDIA SETS SAVE UP TO $44

SPECIAL MIX OR MATCH BONUS
—WHILE SUPPLIES LAST—

You will receive a FREE Simon's Saucer™ package —A $12.95 Premium Value— when your order includes at least 12 items from this Back Issue offer (either 12 media, 12 magazines, or 6 of each).

- A quality, ready-to-run game on cassette tape.
- A durable and attractive ring-binder collector's case for your software library.
- A complete, easy-to-use programming lesson on a deck of colorful flip cards.





MAGAZINES OR MEDIA MAY BE ORDERED SEPARATELY
To Order, Use Order Card On Other Side Of This Page.

Offer & Prices Subject To Change Without Notice.

HOME COMPUTER magazine SHOPPING LIST



BACK ISSUES OF HOME COMPUTER MAGAZINE & MEDIA

	U.S. Surface	CANADA	FOREIGN Surface	FOREIGN Air
Magazine	3.95	5.95	5.50	7.50
Media	6.95	7.95	N/A	10.95
Magazine/Media Set	9.90	12.95	13.95	17.50

SEE PAGES 2 & 3 FOR ALL INDIVIDUAL CONTENTS





PCjr Disk Expansion Kit

Add A Second Disk Drive To Your PCjr for $49.95 PLUS 3.50 SHIP. & HANDL.

The Kit Includes:

- Special Cable To Connect the Second Disk Drive.
- 2 Integrated Circuits.
- A Copy of HCM Vol. 4 No. 4 Containing Instructions.
- Plus An Additional Update Sheet.

—PCjr, Disk Drive, and Power Supply NOT Included—

COMPUTER ACCESSORIES

ITEM NO.	ITEM DESCIPTION	PRICE	SHIPPING 1st one	SHIPPING each add'l
DSK20	20 Blank 5¼" Disks	22.00	3.50	3.50
DSK40	40 Blank 5¼" Disks	42.00	3.50	3.50
DSK60	60 Blank 5¼" Disks	59.40	3.50	3.50
PCJ101	PCjr Disk Expansion Kit	49.95	3.50	—
ODI	ON DISK Revue Vol. 1	14.95	2.00	2.00
	DUST COVERS			
DUS997	10" Monitor	10.95	2.00	.50
DUS72	13" Monior	12.95	2.00	.50
DUS74	MX80 Printer	9.95	2.00	.50
DUS75	Cassette Recorder	4.95	2.00	.50

TEXAS INSTRUMENTS 99/4A CLOSE-OUTS

ITEM NO.	ITEM DESCIPTION	PRICE	SHIPPING 1st one	SHIPPING each add'l
TBK	Best of 99'er	19.95	3.00	3.00
BTPOOL	Best of 99'er—Book & Tape Combo	35.00	ppd./U.S.	—
BPG999	Programmer's Guide	5.00	ppd./U.S.	—
TI0001	99'er Directory	5.00	ppd./U.S.	—
FSN-01	Simon's Saucer	10.95	2.00	2.00
	DUST COVERS			
DUS73	Expansion Box	12.95	2.00	.50
DUS76	TI Console	8.95	2.00	.50
DUS77	Speech Synthesizer	3.95	2.00	.50
DUS78	Disk Box Memory	5.95	2.00	.50
DUS79	32K Memory Exp.	5.95	2.00	.50
DUS80	RS232 Interface	5.95	2.00	.50
DUS81	Disk Controller	5.95	2.00	.50
DUS82	Thermal Printer	8.95	2.00	.50
CBL11	Single Cassette Cable	4.95	1.50	1.50
CBL12	Dual Cassette Cable	4.95	1.50	1.50

Complete the Subscription & Product Order Form and enclose payment or credit card information. Mail To:

Emerald Valley Publishing Co. ● P.O. Box 70288 ● Eugene, OR 97401

Or use our TOLL-FREE Order Line for VISA/MasterCard orders only:

1-800-828-2212 (Minimum $10 Order) In Oregon, Alaska, Hawaii Tel. (503) 485-8796

HOURS: MONDAY THROUGH FRIDAY 7:00AM TO 4:00PM; SATURDAY 9:00AM TO 1:00PM P.S.T.

Defective media gladly exchanged. NO REFUNDS on media. Offer & Prices Subject To Change Without Notice. Satisfaction Guaranteed—or the unfilled portion of your subscription will be refunded, less the cost of any premiums you have received.

BLANK DISK MEDIA OFFER

AS LOW AS 99¢ EACH!

As a service to our readers who prefer to type-in HCM programs, we are able to offer high-quality blank diskettes at very low prices.

	20 DISKS DSK20	40 DISKS DSK40	60 DISKS DSK60
5¼" certified single-sided, double density with reinforced hub ring. Bulk packed with white sleeves and indentification labels.	$22.00	$42.00	$59.40

PLUS $3.50 SHIPPING & HANDLING

BACK ISSUES OF 99'er HOME COMPUTER MAGAZINE & MEDIA

(The Forerunner of HCM)

QTY	MAGAZINE PRICES* U.S.	Canada	Foreign‡	MAG/MEDIA SET PRICES U.S.	Canada	Foreign‡	MEDIA PRICES U.S.	Canada	Foreign‡
3	$7.95	$10.95	$10.95	$15.95	$19.95	$21.95	$10.95	$12.95	$14.95
6	$12.95	$18.95	$18.95	$29.50	$37.95	$39.95	$20.95	$22.95	$25.95
12	$21.95	$29.95	$29.95	$49.95	$59.95	$63.95	$39.95	$44.95	$47.95

‡ Magazines Shipped to Foreign Countries via Surface (Minimum Order of 3)

— Partial Contents —

Issues No. 1-5 OUT OF PRINT

Contents available in book form as "Best of 99'er"—Vol. 1.

See Home Computer Digest or Inquire

THESE 2 MAGAZINES ARE OUT OF PRINT —TAPES AND DISKS ARE STILL AVAILABLE

ISSUE #6 (Partial Contents)
• How To Produce Sound Effects • Debugging a Game Program • How to Start a User's Group • Verbose: A speech Vocabulary Expansion Aid • Color Mapping • Dynamic Manipulation of Screen Character Graphics • The Beginner's Guide to Cassette Operation With the Home Computer • Pre-School Block Letters and Data Compaction • Picking the Ponies in TI BASIC • Battle Star Space Game • 3-D Animation on the Home Computer • Programing Tips • Who is LOGO for? a Tower of Hanoi in TI LOGO • A Review of the TI Lesson-Development Software • An Interview with a Game Designer • Learning Assembly Language with a Magic Crayon • and much, much more.

NOVEMBER 1982 (Partial Contents)
• Chatting with Your Micro: Languages for the Home Computer • A Review of the Smith Corona TP-1 Daisy Wheel Printer • The Micro Jaws Arcade Game • A Knight's Tour in TI BASIC • LOGO Has Style • ASPIC: A Language for Children • A p-System Beginner's Tutorial • An Interview with a p-System Pioneer • A Mini-Memory Screen Dump to the Home Computer Printer • Up Scope!—An exciting Undersea Combat Game • Strategy for Munch Man • A Brief Encounter with a TI Hand-Held Computer • 99'er Shopping Bus • A Pocket Battleship • Sub-Programs in Extended BASIC • Arcade & Adventure Game Reviews • and much, much more.

DECEMBER 1982 (Partial Contents)
• Tex-Scribe: A Text Editor for the Home Computer • A Christmas Computer Carol • Managing a Mailing List the Futura Way • Parsec: The Arcade Game • Plotting With the Home Computer—Pixel by Pixel • Preventing the Situation—Oh No! Memory Full! • A Colorful Tour of TI-Fest: The Home Computer Show • Santa's Workshop: The Making of a Home Computer • The Turtle Arcade: Movies & Video Games in LOGO • Controlling a BASIC Termite • The 99'er Gold Rush—An Arcade/Adventure in the Home • 99'er Digest of News & Happenings in the TI World • Plus Games, Reviews, and much, much more.

JANUARY 1983 (Partial Contents)
• Computer Assisted Instruction for the Handicapped • p-System Basics • Debugging in LOGO • The Dow-4 Gazelle Flight Simulator • Note Whiz and Pitch Master Musical Game Reviews • Learning With the PLATO Computer Library • Strategies for Adventure Gaming • Death Drones • Using the Line-By-Line Assembler • Close Encounters of the Simon Kind • Electrical Engineering Education Program • Interview With an Arcade Game Designer • TI Invaders • Programming With Pascal • Cyber Dice • News and Happenings in the Home Computer World • Arcade Game Reviews • The Thief Adventure Game • Programming Tips • and much, much more.

FEBRUARY 1983 (Partial Contents)
• Texas Instruments at the Winter Consumer Electronics Show • Home Computer Printers on Review • How to Create Math Daisies in LOGO • Vectors in LOGO • ASPIC: A Language for Teachers • The Joys of Adventuring—Part 2 • Programming Pointers with Chuck-A-Luck—Part 4 • Interview With the Voice of Parsec • Why You Need a Printer for Your Home Computer • Lifeline to Titan Space Game • Night Blockade Battleship Game • Tower of Hanoi Pocket Program • Computer Gaming Software Reviews • News of Late Developments in the World of Home Computers • and much, much, more.

MARCH 1983 (Partial Contents)
• An Introduction to the TI-99/2 Basic Computer • The Hex-bus and the 4/A Connection • Making Your Own Say and Spell Game • Disabled Children Learn and Grow • Super Cataloger—A Review of a Disk Library Utility Program • TI's New CC-40 Compact Computer • Robots and Their Social Impact • Twenty Questions With Robot Redford • The Gravity of LOGO • Joystick Jockey—An Overview of Remote Controllers • Parsec Strategy • Converting Extended BASIC to Assembly Language • Matrix Muncher • Mini Memory Disassembler Utility • Pulling the Shade on Sprites • Letters on LOGO • Tiny Tutorials • Games programs, reviews, and much, much more.

APRIL 1983 (Partial Contents)
• Computer Assisted Savings Planning to Build Your Nest Egg • Tex-Cipher Writes and Decodes Secret Messages • Crossbyles—Computer Vocabulary Crossword Puzzle • Cutting Corners On Your Food Budget Using Coupons • Introducing Financial Planning with Multiplan • The Design Philosophy of the Compact Computer • LOGO Takes On the Popular Fifteen Puzzle • Super Language—Programming Sprites in Mini Memory • Colorful Word World—Reading Readiness for Pre-schoolers • Gameware Buffet's A-Maze-ing Boa Alley Game • Giant and Dwarfs Entrapment Game • Game Reviews • Programming Tips • Money Saving Hints • and much, much more.

MAY 1983 (Partial Contents)
• A Consumer's Guide to Word Processing • Word Processing Market Basket • A Generalized Filing Program for VIPs • The Multiplan Medium Balances Your Checkbook and Budget • Activity Accountant Helps School Secretaries with Extracurricular Activities • Maximizing Your Mini Memory's 4K of RAM • Exploring Enhanced BASIC on the Compact Computer • The LOGO Tortoise Debates the BASIC Hare • A Pocket Program to Organize Data with Linked Lists • Mentally Handicapped Learners Team Up with the TI-99/4A • The Wonders of Diskette Storage • Beeline—a Multi-Screen Strategy Game • Lost Ruins—An Archeological Dig Game • 3-D Illusions with Sprites in Depth • Game reviews, Group Grapevine, and much, much more.

JUNE 1983 (Partial Contents)
• Children and Computers Make the 99/4H Connection • Tune Your Guitar with Our TI Tuning Fork • Talk to Your Computer—Voice Technology Is Here • Gameware Buffet's Eat or Be Eaten Aardvark Game • Protect Your Station in the Space Zapper Game • What Multiplan Can and Can't Do • Understanding Inputs and Outputs in Drive For Diskettes—Part 2 • Calculate Loan Schedules on the CC-40 • Go on a LOGO Vacation • Letters on LOGO • A Review of Upper Room Software's Programs for Special Learners • Construct an RS232/Joystick Interface • Group Grapevine • Swapping Bus • A Natural Language Interface for the Professional • Game Reviews • and much, much more.

JULY 1983 (Partial Contents)
• The Evolution of Home Computer Graphics Comes Alive in Graphics Grows Up • Five Data Organizers in Never Out of Sorts • TI & 99'er at the Consumer Electronics Show • WarGames: The Movie and the Book • Editing with Multiplan • The LOGO Logician Presents To Model Is to Learn • LOGO Mosaic Designs Fill the Screen • Your Speech Synthesizer as a Spelling and Foreign Language Teacher • Software for Your Low-cost Printer Port • Gameware Buffet's Treasure Island and the Colorful Switch-A-Row • A Book Review of Learn BASIC for CC-40 Users • 3-D Animation with the TMS 9918A Video Chip • Games Reviews • Group Grapevine • and much, much more.

AUGUST, 1983 (Partial Contents)
• The Home Computer Goes To Work • Bit One, Part Two at the Fashion Factory • Better Business Bar Graphs in Graphic Persuasion • An Ensemble of Assemblers • Cashbox Helps Money Management • Keystrokes for Thrifty Folks—a Review of Typewriter • Game Reviews of Cavern Quest and Starprobe '99 • Counting Fun for Preschoolers • Peripheral Vision 99 • Mean Machines and Small Potatoes • Multiplan Medium Groups Cells into Rectangles • Turtle Text, a LOGO Word Processor • Group Grapevine • Hello, Little Brother—CC-40 Speaks to 99/4A • Plato's Progress Looks at Student Assessment • Pocket Sunrise • Public Investigator • Gameware Buffet's Jungle Jim, Success Formula, and much, much more!

SEPTEMBER, 1983 (Partial Contents)
• Adding on to Your Home Computer • 99'er Directory of Commercially Available Software, Accessories and Peripherals • Peripheral Vision 99 Hardware Reviews • Pocket Sunrise, Part Two in Extended BASIC • Byte Lightning—Mini Memory Plays a Game • TI-WRITER Tutorial • LOGO's Functions, Sets and Turtles • The CC-40 and 99/4A Take the Data, and RUN • PLATO's Progress Looks at Geometry Courseware and the Shape of Things to Come • Gameware Buffet's Challenge of Camelot and The Fly • Game Reviews of Nit Wit and Crime and Punishment • Group Grapevine • 99'er Hall of Fame • 99'er Digest Update on New Products, and much, much more!

OCTOBER, 1983 (Partial Contents)
• Adventures and Fantasy with Your Home Computer • Do-It Yourself Adventure • Picocomputer Emulator, a 4-Bit Microprocessor • Have No Fear: Assembly Language Won't Byte, Part 1 • Make Your Mark • Less Izmore and Debug Computing Cartoon • Once Upon a Tortoise Shell—A LOGO Adventure • Turtle Strut • Number Nibbler for Children • Lots of Plots on Your CC-40 and HX1000 Four-Color Printer/Plotter • Multiplan Medium—Bartender • A Grisly Adventure Bear Hunt • Escape From Wizard's Keep in Extended BASIC • Game Reviews • Shopping Bus • and much, much more.

NOVEMBER, 1983 (Partial Contents)
• Education with your Home Computer • Five Creative Learning Activities for Children • Let's Build America • Have No Fear: Assembly Language Won't Byte, Part 2 • Squeezing the Most Out of TI BASIC • TI-WRITER Tutorial • LOGO Lexicon • Interview with Dale Osborn • TI-WRITER: At Home In the Office • PLATO's Progress • Les Izmore and Debug • The Multiplan Medium • Computer Arrested Instruction—99'er Interviews the Kids • Gameware Buffet: Taco Man and Robo Chase • Game Reviews of Jail Break and Arithmetrex • Hall of Fame • 99'er Digest • and much, much more.

HISTORICAL NOTE
99'er Magazine (founded in December, 1980) was the forerunner of Home Computer Magazine.

THE BEST OF 99'er™

SUPER CLOSE-OUT SPECIAL for TI-99/4A USERS

A Giant Home Computer Compendium™ for the Texas Instruments 99/4A

The largest, most comprehensive collection of programs and articles ever assembled for the TI Home Computer

VOLUME 1

- Over 200 thoroughly tested key-in-and-RUN programs and sub-programs typeset in a grid format for maximum clarity.
- Programming instruction in 4 languages—learn to use BASIC, Extended BASIC, LOGO and Assembly Language—for everything from record keeping and money management to arcade-quality action games.
- A selection of sensational game software featuring full-color graphics, animation, and sound effects.
- Beyond the owner's manual—tips and techniques for getting the most out of your computer system.
- Computer-Assisted Instruction—The home computer becomes your private tutor.
- Page after page of innovative applications—transforming your computer into a home productivity center.

Regular (Pre-Close-out) Prices:

Best Of 99'er (Book alone)	$19.95	+ $3.00 shipping
Best Of 99'er (Tape Set alone)	$35.00	+ $2.50 shipping

SPECIAL OFFER Buy the Tape Set for ONLY $35. And Get the Book FREE + FREE SHIPPING (Canada add $3.00)

PLUS

FREE BONUS WHILE SUPPLIES LAST



ORDER THE BOOK & TAPE PACKAGE AND RECEIVE A Simon's Saucer AND A 99'er Programmer's Guide **ABSOLUTELY FREE!** This Additional $18 gift IS YOURS IF YOU ACT TODAY!



FREE BONUS WHILE SUPPLIES LAST

THE BEST OF 99'er ON TAPE

A Choice Selection of 37 Full-length Programs On 5 Quality Cassette Tapes

- ★ Save Typing Time and Frustrating Key-in Errors.
- ★ Own the Most Comprehensive Software Library for the TI-99/4A
- ★ Enjoy Hundreds of Hours of Exciting Computer Activity.

People Who Know the Home Computer Best™

HOME COMPUTER magazine™

AN EMERALD VALLEY BOOK

EVP

SENSATIONAL SOFTWARE GIVEAWAY

3 FANTASTIC OFFERS

1 YOUR CHOICE OF THIS ISSUE'S PROGRAMS AT A REMARKABLE PRICE—

ONLY $4.95!

plus $1.00 shipping & handling

Price will change to $6.95 upon becoming a "Back Issue."

FOR SUBSCRIBERS & NEWSSTAND PURCHASERS ONLY*

To participate in our monthly Software Giveaway, you need to be a bonafide purchaser of ***Home Computer Magazine***.
You will receive all the programs ON TAPE™ or ON DISK™ (which have versions for your selected machine) whose listings appear in this issue.

IMPORTANT: *Please complete the order form located in the center of this magazine. Enclose it in an envelope along with $5.95 ($6.95 in Canada, $10.95 Foreign Airmail). Payment must be made by check, money order, or VISA/MasterCard. Proof of purchase (subscriber label number, sales receipt or any reasonable facsimile thereof) must also accompany this form.*

**Non-subscriber and non-purchaser price is $12.95 in the U.S.*

SORRY, WE CANNOT ACCEPT TELEPHONE ORDERS FOR THIS SERVICE.



*FREE SOFTWARE!

When Subscribing To

2 HOME COMPUTER magazine

Subscribe or Renew today, and with your paid subscription you will receive **FREE** software—**ON TAPE**™ or **ON DISK**™. With a 1-year subscription to ***Home Computer Magazine*** you get **2 FREE issues** (a $11.90 Premium Value) of **ON TAPE**™ or **ON DISK**™. Subscribe for 2 years and receive **4 issues** (a $23.80 Premium Value). And with a 3-year subscription we'll give you **6 full issues** (a $35.70 Premium Value) of this convenient software on cassette tape or floppy disk.

*DON'T HAVE A COMPUTER? TAKE A RAINCHECK!

We'll give you a raincheck for the FREE software so that when you buy a computer, we will send you your choice of ON TAPE™ or ON DISK™ as a FREE BONUS as stated in this offer.

And You Save Up To 40% Off The Single-Copy Price of the Magazine!

SAVE EVEN MORE!

3 AND ENJOY THE CONVENIENCE OF A PROGRAM SUBSCRIPTION*

By subscribing to **ON TAPE**™ or **ON DISK**™ you will save money off the single-copy price **and** receive the same high-quality programs published in each issue of the magazine—**delivered right to your door!**

This cassette tape or floppy disk program service is the convenient, accurate, and affordable way to save hundreds of typing hours.

The Perfect Addition To Your Magazine Subscription!

- Program Subscribers Get Disks & Tapes First To Coincide With Magazine Arrival.
- Program Subscribers Get Special Updates As Available.

*DOES NOT INCLUDE MAGAZINE

TO ORDER: Please review these 3 Fantastic Offers and use the ORDER FORM located in the center of this magazine.

1 **RECEIVE ALL** the programs in this issue on magnetic media (see below). Indicate your choice of computer media on the order form and enclose $5.95 ($6.95 in Canada, $10.95 Foreign Airmail).
THIS PRICE GOOD FOR THE CURRENT ISSUE ONLY
Proof of purchase must be included with order.

2 **FREE Software** with your subscription to ***Home Computer Magazine*** for the terms listed below:

1-yr (10 issues) $25	2-yr (20 issues) $45	3-yr (30 issues) $63
PLUS 2 FREE	***PLUS 4 FREE***	***PLUS 6 FREE***
ON TAPE or ON DISK	ON TAPE or ON DISK	ON TAPE or ON DISK

Canada add $7 per year; Foreign Surface add $21 for 1-yr magazine subscription. Free software offer available in U.S. & Canada only.

3 **SAVE MONEY** and time with a **Program Subscription*** to **ON TAPE**™ or **ON DISK**™ for the terms listed below:

- 8 ISSUES—*ONLY $44.95* "An Extension To Your 2 Free Issues of Software"
- 10 ISSUES—*ONLY $49.95* "A Full Year Of Program Convenience"

*Does not include magazine.

Canada add $5 for software subscription.
Software subscription not available in other countries at this time.

PLEASE LOCATE THIS BOX ON THE CENTER BOUND-IN ORDER FORM AND INDICATE YOUR CHOICE OF COMPUTER MEDIA

DISK VERSION						CASSETTE VERSION		
APPLE	ATARI	C-64	IBM PC	IBM PCjr	TI	ATARI	C-64	TI

Or ask for a raincheck for your FREE software.

SGV1550885

ALL PROGRAMS IN THIS MAGAZINE



ONLY $4.95* ON DISK OR TAPE!

The same high-quality Apple, Atari, Commodore, IBM, and Texas Instruments programs with type-in-and-RUN listings in this issue are now available ON DISK™ or ON TAPE™ to newsstand purchasers or subscribers of this magazine.

For only $4.95* plus $1.00 for shipping & handling, you receive all the programs for your particular brand of computer—rushed right to your door by First-Class mail.

—Truly A "Software Giveaway!"

For More Information, See ① Inside Rear Cover.

To Order, Use The Bound-In Order Form At Center Of Magazine. Offer & Prices Subject To Change Without Notice.

* Current Single-Issue Price Only — See Center Bound-In Order Form For Back-Issue Prices & Special Program Subscription Offer.